VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

DESIGNING A PROJECT MANAGEMENT WEB APP FOR WORKFLOW OPTIMIZATION AT ALLODIUM GAMES

Bakalářská práce

Autor práce: Daryna Sharpata

Vedoucí práce: Mgr. Jana Valentová

Jihlava 2026

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Daryna Sharpata**
Studijní program:	Aplikovaná informatika
Garant studijního programu:	doc. Ing. Lenka Kuklišová Pavelková, Ph.D.
Název práce:	**Designing a Project Management Web App for Workflow Optimization at Allodium Games**
Vedoucí práce:	Mgr. Jana Valentová
Cíl práce:	To develop a user-centered design for a project management web application for Allodium Games, focusing on key features such as dashboards, time tracking, and reporting. The design will be created in Figma through an iterative process based on user research and validated through testing with stakeholders to ensure optimal workflow efficiency and usability. The final deliverable will include a high-fidelity prototype and a comprehensive design system, which will be implemented into the company's operations.

Abstrakt

Bakalářská práce se zabývá návrhem uživatelsky orientované webové aplikace ClickDown pro společnost Allodium Games, herní studio pracující s externími freelancery. Cílem práce bylo identifikovat konkrétní slabá místa v interních procesech sledování pracovní doby a vyřešit je prostřednictvím dedikovaného reportovacího nástroje. Na základě rozhovorů s technickým ředitelem a účetní byly definovány klíčové požadavky systému. Návrh postupoval metodou uživatelsky orientovaného designu dle normy ISO 9241-210 a zahrnoval tvorbu informační architektury, uživatelských toků a nízkoúrovňových drátěných modelů. Ověření proběhlo prostřednictvím testování použitelnosti metodou Wizard of Oz se třemi účastníky. Zjištěné problémy byly zapracovány do vysokoúrovňového prototypu podloženého uceleným designovým systémem. Výsledkem je vysokoúrovňový prototyp a kompletní designový systém připravené k vývojové implementaci.

Klíčová slova

UX/UI design; webová aplikace; uživatelsky orientovaný design; reportování pracovní doby; použitelnost

Abstract

Bachelor thesis presents the design of a user-centred web application, ClickDown, developed for Allodium Games, a game studio operating with distributed freelance workers. The aim was to identify specific bottlenecks in the studio's internal time-tracking and reporting processes and to resolve them through a dedicated reporting tool. Stakeholder interviews with the chief technology officer and the company accountant informed the system requirements. The design followed the human-centred design methodology defined by ISO 9241-210 and encompassed information architecture, user flows, and low-fidelity wireframes. The wireframes were evaluated using a Wizard of Oz usability test with three participants. Findings were incorporated into a high-fidelity prototype supported by a comprehensive design system. The result is a fully specified, tested design prepared for development implementation.

Keywords

UX/UI design; web application; user-centred design; time reporting; usability testing

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „**AZ**“).

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle Směrnice pro vedení, vypracování a zveřejňování závěrečných prací na VŠPJ, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 12. dubna 2026

…….

Podpis studenta/ky

Acknowledgements

I would like to thank my family and friends for their support and patience throughout this process. I also appreciate the teachers and the environment at VŠPJ, which gave me space to learn, explore, and develop my skills.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
CSV	Comma-Separated Values
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
SSO	Single Sign-On
UCD	User-Centred Design
UI	User Interface
UX	User Experience

Introduction

The digital games industry increasingly relies on distributed teams of freelance workers whose time must be tracked, validated, and reported across multiple client projects. For independent studios, the ability to generate accurate, consolidated reports is not only a matter of operational efficiency but a prerequisite for financial transparency and reliable invoicing. At Allodium Games, the need to improve project management workflows formed the starting point for the work.

The goal of the thesis is to design a user-centred web application that resolves the specific workflow inefficiencies identified at Allodium Games. Following the human-centred design process defined by ISO 9241-210 (2019), the work began with a structured analysis of existing workflows rather than with predefined design assumptions. Interviews with the chief technology officer and the company accountant were conducted to understand the current situation from the perspective of the primary stakeholders.

The analysis revealed that core project management functions, including task assignment, progress tracking, and team coordination, were already functioning adequately within the studio's existing ClickUp environment. The primary source of workflow friction was located at the intersection of time tracking and financial reporting. Freelance workers lacked a reliable interface for verifying their monthly hours, and the accountant depended on a manual, fragmented process to generate the consolidated summaries required for invoicing and payroll. No existing tool within the company's infrastructure addressed the specific combination of data aggregation and report structure needed.

The outcome of the thesis is ClickDown, a custom web application designed as a dedicated reporting layer built on top of the ClickUp API. Rather than replacing already operational infrastructure, ClickDown targets the precise point of friction identified through research, delivering workflow optimisation where it is most needed. The application was developed through an iterative user-centred process and validated through usability testing with representative participants.

Work is structured in two main parts. The theoretical part analyses existing project management and reporting platforms, identifies recurring usability gaps in time reporting, and establishes the conceptual framework underpinning the design process. The practical part documents the full design cycle: stakeholder interviews, requirements specification, low-fidelity wireframing, usability testing, and the development of a high-fidelity prototype supported by a complete design system.

1 Theoretical part

The theoretical part establishes the foundation for the design of ClickDown, a project management reporting tool developed for Allodium Games. The chapter is structured in two parts. The first part analyses existing project management and time reporting platforms, specifically ClickUp, Jira, Asana, and Harvest, with a focus on their reporting capabilities, dashboard features, and limitations in supporting individual and organisational time visibility. The analysis identifies recurring usability gaps that motivate the development of a tailored solution. The second part defines the theoretical framework underpinning the design process, drawing on principles of User-Centred Design, interface usability, data visualisation, and technical integration constraints. Both areas together provide the conceptual and methodological basis on which the practical design work presented in Chapter 2 is grounded.

1.1 Project management tools and reporting systems

Efficient project management tools have become a cornerstone of modern software development companies, enabling teams to coordinate tasks, track progress, and optimize workflows. A critical aspect of project management systems is their ability to provide transparent and reliable reporting. Without adequate reporting functionality, organizations face inefficiencies, particularly when validating work hours or preparing financial documentation such as invoices.

At Allodium Games, ClickUp is currently the central project management tool. While it provides a wide set of features, its reporting capabilities are limited and poorly aligned with the company's needs. To justify the development of a tailored solution, it is necessary to analyse existing project management systems, focusing specifically on their time reporting and dashboard functionalities.

1.1.1 Overview of project management apps (ClickUp, Jira, Asana, Harvest)

ClickUp is an all-in-one productivity and project management platform that enables teams to plan, organize, and collaborate on tasks within a unified workspace. The tool was developed to simplify and centralize project processes across industries, offering functionality for task management, documentation, communication, and performance tracking (ClickUp, 2024m). The system supports more than fifteen different visual task views, including List, Board, Calendar, Gantt, and Timeline views, allowing teams to tailor their workspace to specific methodologies such as Agile or Kanban. Collaboration is embedded into every feature, enabling real-time communication, document sharing, and progress tracking across teams of any size.

ClickUp's flexibility extends beyond task management. It provides a comprehensive dashboard system designed to visualize productivity metrics and performance indicators in one place. Users can create custom dashboards from various data sources, integrating charts, progress bars, tables, and formula-based widgets to monitor individual or team progress (ClickUp, 2024b). The dashboards allow users to measure and analyse metrics such as project completion, resource allocation, and workflow efficiency. Advanced features such as ClickUp Brain serves as the integrated AI layer. It helps users query dashboard data in natural language, enhancing accessibility and speed of insight (ClickUp, 2024b).

ClickUp's wide functionality and configurability make it a robust choice, yet the same features can introduce challenges for organizations seeking simplicity and clarity. For example, Allodium Games primarily requires a straightforward time reporting solution that aggregates tracked hours for specific time ranges, rather than complex analytics or multiple metric types. In such contexts, the extensive customization options and large number of features may create unnecessary cognitive load for users and reduce usability.

Jira, developed by Atlassian, is an industry-leading project management and issue-tracking platform that enables teams to plan, monitor, and deliver work collaboratively. Originally designed for software development, it has evolved into a universal tool adaptable for diverse teams including marketing, design, and operations (Atlassian, 2025a). Jira structures projects through “issues” and “projects,” allowing users to organize, prioritize, and track tasks with precision. It supports multiple methodologies – most notably Agile, Scrum, and Kanban – through customizable boards, timelines, and workflows. Both features provide transparency across project stages and facilitate coordination between technical and non-technical teams.

Beyond task management, Jira provides a comprehensive reporting and analytics system to help teams evaluate progress and performance. Its reporting capabilities are categorized into four types: Agile reports (focusing on sprint velocity and performance trends), DevOps reports (for deployment pipelines and release frequency), issue analysis reports (for workload distribution), and forecasting or management reports (for capacity and planning) (Atlassian, 2025b). Dashboards in Jira consist of configurable “gadgets,” which allow users to monitor multiple data sources in real time. The “gadget” can display charts, summaries, and filters, and may be private or shared with other users. While the dashboards support advanced customization, their complexity can be overwhelming for organizations requiring only simplified, high-level time summaries.

Asana is a web-based project management platform that helps teams plan, organize, and track work in a shared environment. It structures tasks within projects, portfolios, and goals, enabling users to visualize progress through multiple views such as lists, boards, calendars, and timelines. Both features make it suitable for managing both strategic initiatives and day-to-day operations (Asana, 2025a).

Reporting in Asana is designed to make project data more transparent and actionable. The Universal Reporting feature allows users to build customizable dashboards that visualize progress across projects and teams. Reports can display metrics such as task completion rates or workload distribution, and can be filtered by user, project, or time range. The feature helps organizations identify productivity trends and monitor overall workflow efficiency (Asana, 2025b).

Asana provides a clear and visually intuitive reporting system, yet it focuses mainly on general project insights rather than detailed time tracking. For companies such as Allodium Games, the limitation reduces its suitability as a primary reporting tool for time-based analysis.

Harvest is a lightweight, cloud-based time tracking and invoicing tool designed to help teams record, analyse, and bill their work efficiently. It provides a straightforward way to log hours across browsers, desktop, and mobile applications, ensuring that data collection is simple and consistent. Core features include flexible time entry options (live timers or manual input), automatic reminders to maintain tracking accuracy, and integration with popular productivity

tools such as Slack and Trello. Its reporting functions visualize time distribution, project budgets, and team capacity; while invoicing and online payment tools (via Stripe and PayPal) streamline client billing (Harvest, 2025).

The system's reporting module enables users to generate customizable summaries of time usage, project costs, and performance. Reports can be exported and shared automatically, supporting management in identifying workload imbalances or tracking budget adherence. Although Harvest offers an intuitive user experience and strong financial integration, its focus on billable versus non-billable hours and profit tracking makes it more suitable for client-service businesses than for internal development teams.

At Allodium Games, Harvest was previously used for time tracking purposes. Over time, however, the software proved insufficient for the company's needs. The reporting interface was found to be too limited and the overall user experience unsatisfactory. As a result, the organization transitioned its project and time management processes to ClickUp, seeking tighter integration between tasks and tracked time. Nevertheless, the need for a clear, customizable overview of working hours and task activity, which are crucial for invoice preparation and internal analysis, remained unresolved. Because ClickUp's data structure and reporting tools did not provide the required transparency, Allodium now maintains synchronized data in its own database. Developing a dedicated internal web application therefore represents a logical step toward achieving greater control, reliability, and efficiency in reporting workflows.

1.1.2 Analysis of time reporting and dashboard features

ClickUp dashboard system enables teams to visualize performance indicators through customizable widgets (known as cards) that display data such as workload, progress, and tracked hours (ClickUp, 2024b). Dashboards can be attached to specific workspaces, folders, or lists depending on the organizational structure, allowing for flexible data grouping. As shown in Figure 1, ClickUp dashboards combine multiple visualization types – such as bar charts, pie charts, and numerical summaries – into one interface for real-time monitoring of tasks and productivity.

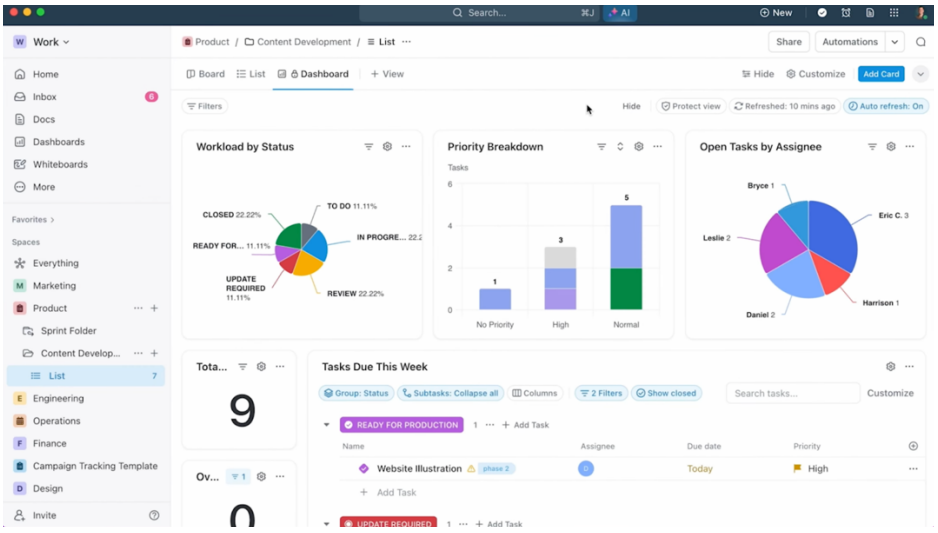

Figure 1: ClickUp Dashboard interface displaying project workload and task progress

Source: ClickUp (2023)

ClickUp dashboards are highly configurable and well suited for managerial oversight, yet less intuitive for individual contributors who need personalized overviews of their work hours. Filtering and adjusting widgets to display only one user's data requires multiple manual steps, and the system lacks a predefined per-user summary. The complexity makes the reporting experience less accessible for non-technical users and limits its usefulness for quick, individual time analysis (ClickUp, 2024b).

ClickUp also provides a Timesheet view, which forms part of its time reporting functionality. The Timesheet interface allows freelance workers to log and review their tracked hours for specific tasks in a more personalized context. As illustrated in Figure 2, the Timesheet only displays data on a weekly basis and does not generate a cumulative monthly overview.

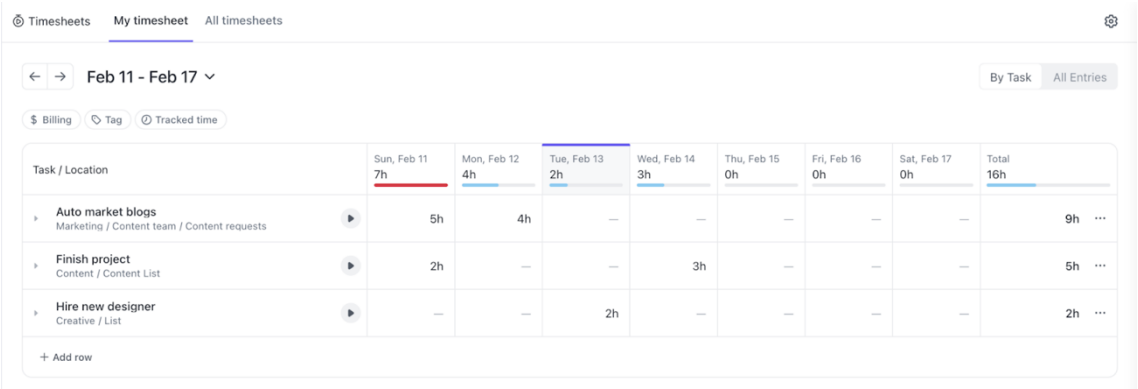

Figure 2: ClickUp Timesheet view showing weekly tracked hours per user

Source: ClickUp (2024j)

Users must therefore manually calculate totals when preparing invoices or reviewing their overall productivity. The design limitation highlights a gap in ClickUp's reporting ecosystem. Dashboards serve managers effectively, yet there is no equivalent user-friendly tool for individuals requiring long-term visibility of their working hours.

Jira offers an extensive set of reporting and dashboard features that allow users to visualize project data and monitor productivity. Its dashboards consist of configurable elements called gadgets, which can display metrics such as sprint progress, issue statistics, or team workload. The dashboards can be personalized and shared across projects, providing flexibility for team-level performance tracking. As shown in Figure 3, Jira dashboards are modular and data-rich, presenting comprehensive views suitable for agile project management and performance evaluation (Atlassian, 2025a).

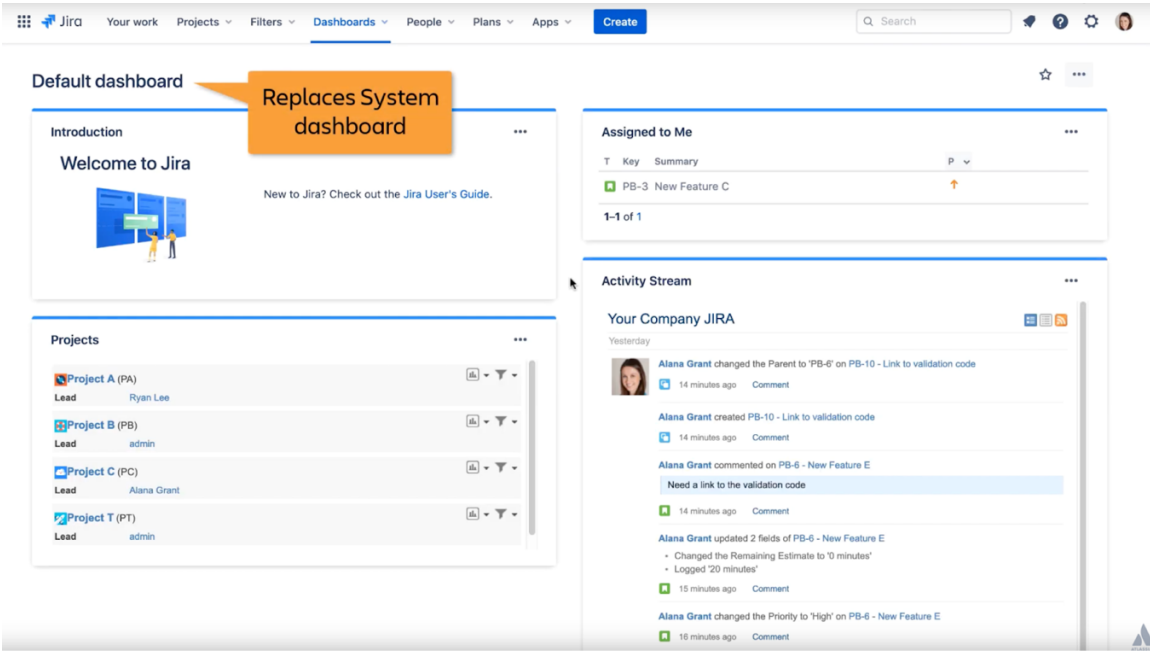

Figure 3: Jira dashboard interface displaying customizable reporting gadgets

Source: Atlassian (2025a)

Despite their flexibility, Jira's dashboards are primarily oriented toward managerial oversight rather than individual user reporting. While they effectively display team metrics, they lack built-in options for simple per-user time summaries. Time tracking in Jira occurs at the task level, and users must manually log the hours spent on each issue. The native Time Tracking Report allows managers to compare estimated versus actual work, but it does not aggregate total hours per employee or provide a clear overview for longer timeframes such as months (Atlassian Community, 2025). Figure 4 shows Jira's built-in time tracking report, which lists time logged per issue but lacks aggregated user or monthly views.

Figure 4: Jira native time tracking report showing estimated and logged work per task

Source: Atlassian Community (2025)

To obtain more detailed reports, users often rely on external plugins such as Worklogs – Time Tracking and Reports, which extend Jira's functionality with features like grouping by project or user and visual summaries of logged hours. The integrations improve usability but introduce additional complexity and dependency on third-party tools. Jira excels in high-level project analytics, yet it does not provide a cohesive, user-centered approach to individual time reporting. From a UX perspective, readability takes a lower priority than configuration and technical control, which contrasts with the user-oriented design principles applied in the present work.

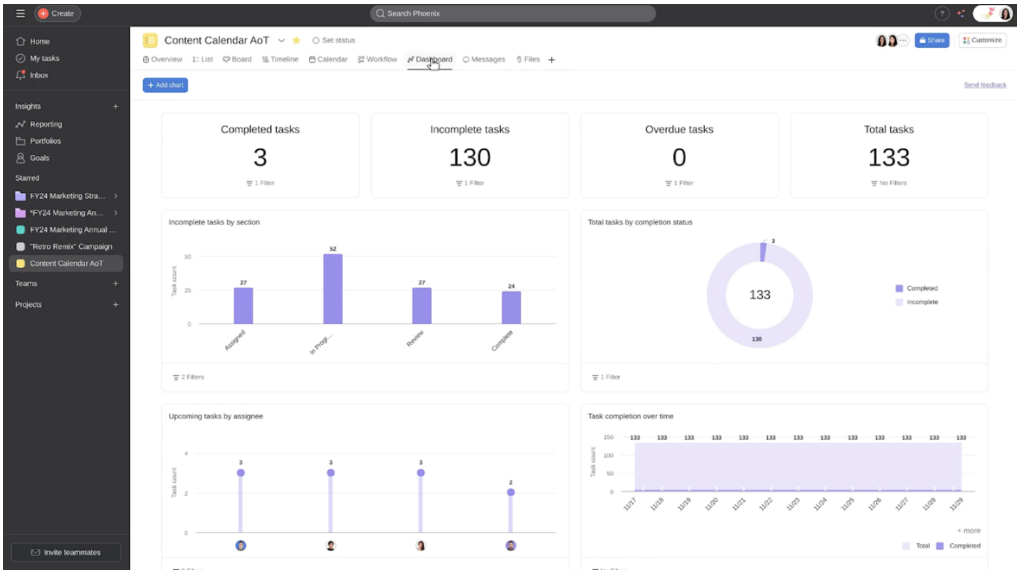

Figure 5: Asana dashboard interface displaying portfolio-level reporting charts

Source: Asana (2025a)

Asana integrates dashboards and time tracking to provide visual insights into project progress and performance. Dashboards are built from customizable charts and metrics that convert project data into visual summaries. Users can create dashboards at the project or portfolio level, as illustrated in Figure 5, allowing managers to monitor multiple teams and compare workloads and progress in real time (Asana, 2025a).

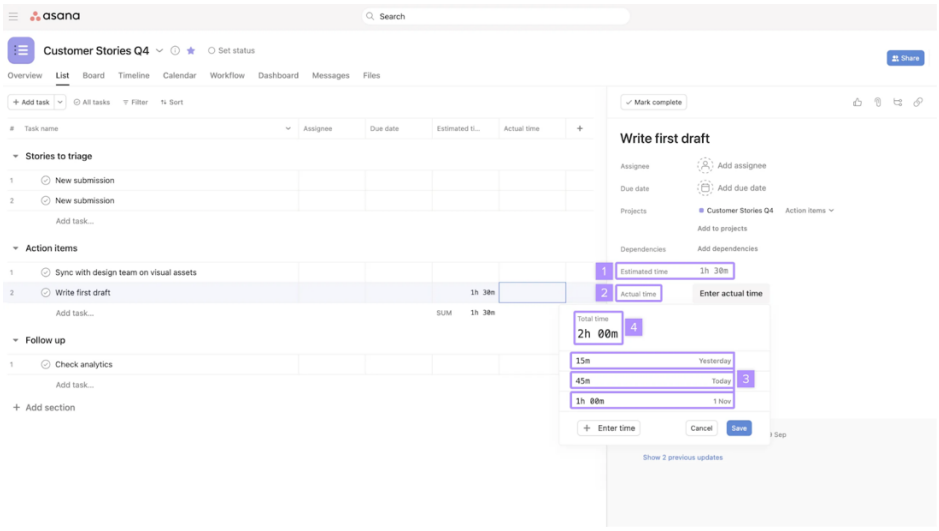

Figure 6: Asana time-tracking interface showing actual hours logged per task

Source: Asana (2025b)

While dashboards are flexible and visually clear, their effectiveness depends on well-structured custom fields such as “Estimated Time” or “Actual Time.” The fields form the basis for Asana’s time reporting; however, in many companies, estimated time is not used as a formal metric, since task duration and tracking are often managed directly by users. Figure 6 shows Asana’s time-tracking interface, where users record actual worked hours per task through custom fields (Asana, 2025b).

Asana supports real-time tracking and data export yet generating monthly or per-user reports requires manual configuration. The absence of predefined templates or aggregated summaries makes it less suitable for organizations needing simple overviews for invoicing or productivity analysis. From a UX perspective, Asana’s design prioritizes flexibility and visualization over straightforward personal reporting — a limitation consistent with the broader challenges of existing project management tools analysed in the present thesis.

Harvest functions primarily as a time-tracking and invoicing tool rather than a project management platform. Its interface focuses on simple timesheets and timers, allowing users to record hours by project or client and view weekly summaries (Figure 7). Reporting features enable filtering by client, project, or time range and export to CSV, but visualization options remain minimal. The only dashboard available is the *client dashboard*, which provides a high-level overview of hours and budgets without deeper task-level analysis (Harvest, 2025).

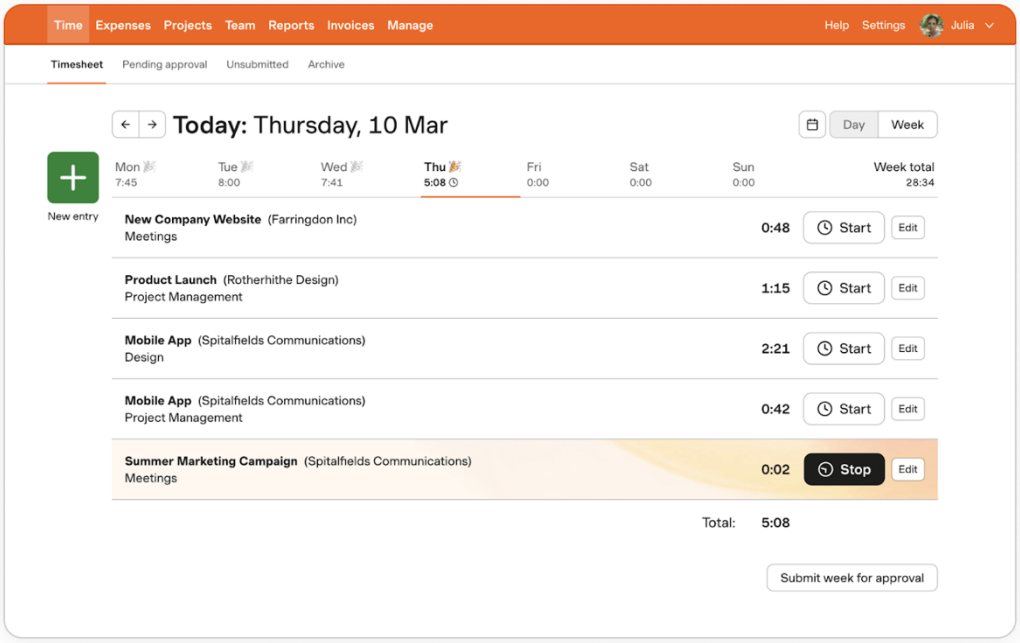

Figure 7: Harvest time-tracking interface showing weekly timesheet entries
Source: Harvest (2025)

Although efficient for personal or small-team use, Harvest lacks integration with complex project structures. In practice, its simplicity limits workflow transparency when managing multiple projects simultaneously.

The company for which the new solution is being developed previously used Harvest to record working hours and prepare invoices. However, because time entries existed separately from

project data managed in other tools, freelance workers needed to duplicate effort when reviewing tasks and tracked time. The separation reduced efficiency and highlighted the need for an integrated reporting approach. From a UX standpoint, Harvest demonstrates the usability benefits of simplicity but also reveals the limitations of standalone tracking systems that lack contextual data.

1.1.3 Common weaknesses in existing project and time reporting tools

A comparative analysis of the reviewed tools reveals that, although each system provides effective mechanisms for tracking and visualizing time, they all exhibit recurring limitations that reduce their usability in integrated workflow environments. The weaknesses primarily concern data export limitations, usability complexity, and restricted customization options, which affect both everyday users and management.

The main shortcomings of each analysed application are summarized in Table 1.

Table 1: Comparative overview of reporting limitations in selected tools

System	***Main Limitations***	***Impact on Usability***
ClickUp	Complex dashboards; no automatic per-user summary; requires manual filtering for individual reports.	Increases time needed for users to verify monthly work hours and prepare invoices.
Jira	Requires technical setup (JQL) for user-specific or long-term reports; lacks aggregated time views.	Limits accessibility for non-technical users; discourages use of native time reporting.
Asana	Depends on manual creation of custom fields; no preconfigured time report templates; lacks monthly summaries.	Demands additional setup effort; inconsistent user experience across teams.
Harvest	Lacks project integration and contextual task data; provides only basic numeric reporting; minimal visual insights.	Reduces analytical value and cross-project visibility; suitable only for simple individual tracking.

Source: own elaboration (2025)

As shown in Table 1, despite differences in technical implementation, similar usability patterns emerge across all platforms. Systems designed for large-scale project management (ClickUp, Jira, Asana) prioritize flexibility and configurability, often at the expense of clarity and simplicity for individual contributors. Lightweight tools such as Harvest emphasize ease of use but fail to deliver contextual insight or support complex reporting needs.

Another significant issue concerns data export and interoperability. Most tools allow CSV or API exports, yet the exported data is often fragmented and requires post-processing to produce coherent monthly or project-level reports. The limitation creates inefficiency in organizations that rely on time-tracking data for billing or performance evaluation.

From a UX perspective, the absence of unified reporting templates or automatically generated personal summaries represents a weakness shared by all reviewed tools. Users are forced to perform repetitive filtering and manual calculations, increasing the likelihood of human error. The friction undermines the primary goal of time reporting – to provide transparent, accurate, and easily interpretable data.

Modern project management platforms offer extensive features for planning and collaboration, yet they lack a user-centered approach to time reporting that would simplify daily operations for freelance workers while supporting managerial oversight. The recurring gap establishes the foundation for developing a new solution that combines ClickUp data integration with intuitive, customizable, and visually coherent reporting – objectives central to the design process presented in the following chapters.

The analysis also confirmed that Allodium Games already uses ClickUp for core project coordination. The primary unmet need is therefore not a replacement for existing tooling, but a dedicated reporting layer that addresses the specific financial and transparency gaps identified across all reviewed platforms.

1.2 Theoretical foundations for designing a user-centred reporting solution

Designing an effective reporting interface requires a clear theoretical basis that connects user needs, data visualization, and system functionality. Following the analysis of existing project management tools, the section defines the conceptual framework that supports the creation of a user-centred reporting solution. The focus lies on the principles of user-centred design, interface usability, and information clarity, which are the key aspects ensuring that reporting tools provide both accuracy and accessibility.

Since the proposed system integrates data from external sources, it is also essential to understand how interface design depends on data structure and technical constraints such as APIs. The following chapters therefore outline the design methodology, UX/UI principles, and data-related factors that form the foundation for the proposed reporting interface.

1.2.1 Design methodology framework

The methodological foundation for the proposed reporting interface follows the principles of User-Centred Design (UCD) as defined by the international standard ISO 9241-210 (2019). UCD establishes that design should be based on an explicit understanding of users, their tasks, and the environments in which a product is used. It also requires active user involvement, iterative development, and evaluation at each stage of the process. The principles ensure that the final solution is not only functional but also aligned with real user behaviour and expectations.

According to ISO 9241-210, the human-centred design process consists of four key activities:

(1) understanding and specifying the context of use,

(2) specifying user and organisational requirements,

(3) producing design solutions, and

(4) evaluating designs against requirements.

The cyclical structure forms the conceptual backbone of the thesis, as it encourages continuous feedback loops between users and designers, preventing usability problems from appearing in later stages. The process is typically visualised as shown in Figure 8, where each activity leads iteratively to the next, allowing evaluation and redesign at any stage.

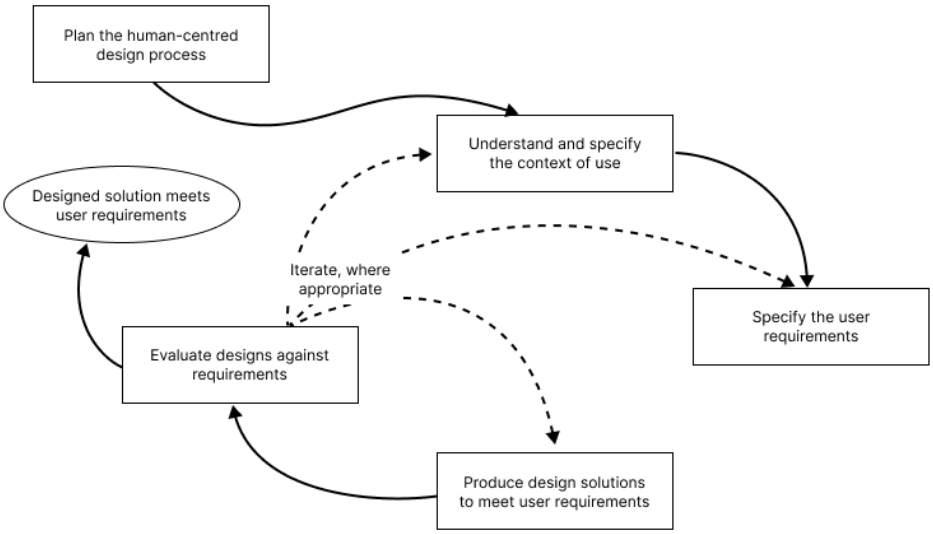

Figure 8: Human-centred design process for interactive systems
Source: own elaboration based on ISO 9241-210 (2019, p. 11)

The ISO 9241-210 standard establishes a structural framework for human-centred design, and Nielsen's (1993) "Usability Engineering" model complements it with concrete methods for evaluating and refining user interfaces. His approach emphasises iterative prototyping, heuristic evaluation, and small-sample usability testing as efficient means of identifying usability issues early and at low cost. Integrating Nielsen's lifecycle into the ISO framework creates a pragmatic, evidence-driven process particularly suitable for small-scale internal projects such as the ClickDown reporting tool. The iterative nature of Nielsen's model, characterised by the continuous alternation between design, implementation, and evaluation, closely parallels the cyclical structure of the User-Centred Design (UCD) process.

Table 2 summarises the stages of Nielsen's usability engineering lifecycle, which collectively outline how user analysis, prototyping, and empirical testing form an integrated, iterative process aimed at achieving measurable usability outcomes.

Table 2: The stages of the usability engineering lifecycle model

Stage	***Description***
1. Know the user	Analyse user characteristics, tasks, and goals
2. Competitive analysis	Evaluate similar products and market standards
3. Setting usability goals	Define measurable usability objectives
4. Parallel design	Develop multiple design alternatives
5. Participatory design	Engage users in the design process
6. Coordinated design of total interface	Ensure consistency across UI components
7. Apply guidelines and heuristic analysis	Use established usability principles
8. Prototyping	Create and test design prototypes
9. Empirical testing	Validate designs with real users
10. Iterative design	Refine based on evaluation feedback
11. Collect feedback from field use	Monitor usability in real-world conditions

Source: Nielsen (1993, p. 72)

To maintain consistency and clarity in interaction, the present work draws upon Norman's (2013) fundamental principles of design: affordances, signifiers, mappings, feedback, constraints, and conceptual models. Together, the principles determine how users perceive and interact with a system, forming the foundation of effective and discoverable design. Applying them to dashboard and reporting interfaces helps ensure that users immediately understand available actions and system responses. For example, when filtering time ranges or exporting data. The principles reduce cognitive load and enhance the perceived transparency of the reporting process.

A concise summary of Norman's principles is provided in Figure 9, which illustrates their role within the interaction design layer of the reporting interface.

Figure 9: Core interaction design principles

Source: own elaboration based on Norman (2013, pp. 13–25)

In parallel, Design Thinking provides an overarching creative mindset that complements UCD. Brown (2009) describes Design Thinking as an iterative, human-oriented approach balancing analytical reasoning with intuitive exploration. Incorporating elements of the methodology supports ideation and exploration of alternative reporting layouts before high-fidelity prototypes are finalised.

The framework distinguishes between "usability" and "user experience" following the UX White Paper (Roto et al., 2011). Whereas usability focuses on efficiency and effectiveness, UX considers users' emotional responses and long-term satisfaction. The distinction is relevant for internal reporting systems, where the aim extends beyond functional task completion toward creating a sense of clarity, control, and confidence when reviewing or validating tracked time.

A comparative summary of all theoretical approaches is presented in Table 3, outlining their key focus areas and application within the present work.

Table 3: Comparison of theoretical frameworks applied in the thesis

Framework	***Main focus***	***Application in the thesis***
ISO 9241-210 (UCD)	Iterative, user-focused process	Structural framework for design and evaluation
Nielsen (1993)	Practical usability testing	Heuristic evaluation and discount usability testing
Norman (2013)	Interaction design principles	Used in interface and component design
Brown (2009)	Creative problem-solving mindset	Supports ideation and exploration of layout variants
Roto et al. (2011)	Broader user experience evaluation	Ensures emotional clarity and perceived transparency

Source: own elaboration based on ISO 9241-210 (2019); Nielsen (1993); Norman (2013); Brown (2009); Roto et al. (2011)

In summary, the theoretical model guiding the present work combines ISO's structured UCD process, Nielsen's evaluative methods, Norman's interaction principles, and the experiential dimension defined in UX literature. Together, they establish a rigorous methodological foundation for developing an intuitive, reliable, and user-validated reporting interface within the broader project management ecosystem.

1.2.2 Principles of UX/UI design for reporting interfaces

Designing reporting interfaces requires a balance between aesthetic clarity and functional usability. General UX principles focus on making interaction efficient and satisfying (ISO 9241-210, 2019), yet dashboards and reporting tools involve an additional challenge: ensuring that data is not only accessible but also comprehensible at a glance. According to Few (2006), a dashboard should present the most relevant information "on a single screen so the information can be monitored at a glance," which highlights the importance of concise communication, perceptual clarity, and cognitive efficiency. Described ideas form the theoretical foundation for the design of the ClickDown reporting interface, whose purpose is to present time-tracking data from ClickUp in a visually coherent and usable way.

Clarity and information hierarchy are essential to achieving the goal. Few (2006) argues that visual hierarchy should guide attention toward the most significant metrics while maintaining logical grouping and spatial alignment. Gestalt principles, which are proximity, similarity, and continuity, help users quickly recognize patterns and relationships between data elements (Few, 2006, pp. 74–79). In the context of ClickDown, the principles outlined above will inform the layout of time-reporting modules, ensuring that summaries such as total hours or project-based distributions stand out without overwhelming the user. The structure will therefore support both individual freelance workers, who need quick monthly overviews, and managers, who monitor broader trends across teams.

Visual simplicity represents another key factor in dashboard design. Few (2006) emphasizes the need to remove unnecessary decorative elements, focusing on what he terms the "data-ink ratio," where every visual element contributes meaning. Nielsen (1993) similarly identifies

minimalist design as one of his ten usability heuristics. For ClickDown, described principle supports a clean interface that communicates information without distraction, using neutral colours, restrained typography, and consistent spacing to enhance data readability. Norman's (2013) concept of constraints further reinforces the approach by preventing interface clutter and limiting user actions to what is contextually meaningful.

Consistency also plays a crucial role in usability and user trust. When visual and interactive patterns are predictable, users can form accurate mental models of system behaviour (Norman, 2013). In reporting tools, mentioned principles applies to the consistent placement of filters, charts, and date selectors across pages (Few, 2006, p. 143). ClickDown will apply principle outlined above by using unified design components and behaviour patterns across dashboards and tables, ensuring that users can transfer their understanding from one module to another without confusion.

Feedback and interaction transparency represent another essential dimension of reporting interface design. Norman (2013) states that clear, perceivable feedback after user actions enhances trust and comprehension. In a reporting interface, actions such as changing a date range or exporting data should immediately update the visible results, providing assurance that the system has registered the input. Few (2006) recommends small visual cues – such as colour changes, highlights, or motion – to confirm user actions without interrupting their workflow. For ClickDown, implementing the micro interactions will make the experience more intuitive and transparent.

Another dimension concerns perceptual design and colour usage. Few (2006) highlights the importance of encoding information through colour and form in a way that leverages preattentive visual processing. Limiting the palette and using contrast purposefully enhances quick data interpretation—for example, distinguishing completed, active, and pending projects. Outlined approach aligns with Norman's (2013) notion of feedback and supports visual consistency across data displays. In ClickDown, semantic colour coding will assist both freelance workers and financial analysts in distinguishing categories of tracked work at a glance.

Accessibility and readability must also be maintained across devices and viewing contexts. Following ISO 9241-210's human-centred principles, the design must accommodate diverse users by ensuring sufficient contrast, appropriate text sizes, and responsive layouts. Incorporating standards such as the Web Content Accessibility Guidelines (WCAG 2.1) ensures that users with visual differences can still interpret the information accurately. In the ClickDown dashboard, readability will be prioritised through simple typography and responsive alignment that adapts to different screen sizes.

Table 4: Key UX/UI principles for reporting interfaces and their application to ClickDown

Principle	*Definition / Theoretical basis*	*Application in ClickDown design*
Clarity & Information Hierarchy	Visual hierarchy directs user attention toward the most relevant data while maintaining logical grouping and alignment (Few, 2006).	Layout prioritises total hours and project summaries, using grouping and alignment to simplify interpretation.
Visual simplicity (data-ink ratio)	Non-essential decoration should be reduced to prevent cognitive overload; design should prioritise meaning over form (Few, 2006; Nielsen, 1993).	Neutral colour palette, minimal icons, and clear typography enhance data readability and reduce distraction.
Consistency & predictability	Consistent patterns support learnability and user trust by reinforcing mental models (Norman, 2013).	Uniform placement of filters, buttons, and charts across modules improves familiarity and navigation.
Feedback & interaction transparency	Immediate feedback after user actions confirms system responsiveness and enhances user control (Norman, 2013).	Micro-interactions (hover highlights, update animations) signal changes when users filter or export data.
Perceptual design & colour use	Effective colour contrast and semantic mapping support rapid interpretation and emotional coherence (Few, 2006).	Colour coding distinguishes project states and user categories, enhancing quick pattern recognition.
Accessibility & readability	Usability depends on inclusivity: clear text, sufficient contrast, and responsiveness support all users (ISO 9241-210, 2019).	WCAG-compliant colours and scalable typography maintain clarity across devices and lighting conditions.

Source: own elaboration based on Few (2006); Norman (2013); Nielsen (1993); ISO 9241-210 (2019)

The UX/UI principles most relevant to reporting interfaces combine perceptual clarity, visual consistency, and interactive transparency. Drawing on the frameworks of Few (2006), Norman (2013), Nielsen (1993), and ISO 9241-210 (2019), the design of ClickDown aims to transform complex time-tracking data into an interface that supports understanding, reduces cognitive load, and improves decision-making. Described principles collectively ensure that the final product will not only present information effectively but also reflect the values of usability, clarity, and inclusivity central to human-centred design.

1.2.3 Data and Integration as Design Constraints

In digital product design, technical infrastructure acts not only as a medium of implementation but also as a determinant of what can be designed and experienced. According to Maguire (2001), human-centred design requires the designer to understand both the context of use and the technological environment in which interaction occurs. In the case of ClickDown, the environment is defined by the ClickUp API, its data model, and the synchronization logic that governs how information is retrieved, processed, and displayed. The structural and integration constraints directly influence the form, responsiveness, and usability of the reporting interface.

APIs (Application Programming Interfaces) function as bridges between data systems, enabling applications to exchange information in a structured and secure way (ClickUp, 2024m). In UX

terms, they define the range of possible interactions that can be visually represented (Nielsen, 1993). The ClickUp API follows a REST architecture, using standardized methods (GET, POST, PUT, DELETE) to manipulate resources such as tasks, lists, and users. The authentication process relies on access tokens that verify permissions, ensuring data privacy while limiting the frequency of requests according to rate-limit policies (ClickUp, 2024a; ClickUp, 2024h). The technical characteristics set the foundation for ClickDown's back-end communication, where each user's time-tracking data, project hierarchy, and task information are fetched and processed for visualization.

The design of the reporting interface must therefore align with the underlying data model. In ClickUp, the hierarchical structure (Workspace → Space → Folder → List → Task → Subtask) is supplemented by custom fields and time-tracking entries that store contextual information such as project name, client, duration, and date (ClickUp, 2024i; ClickUp, 2024d). Each time entry is represented as a record linked to a specific task and user, identified by a unique ID and timestamp in UTC format (ClickUp, 2024e). The fields define not only what can be displayed in ClickDown but also how filtering, sorting, and summarization are designed in the interface. For instance, since time entries are retrieved as individual objects, aggregate values – such as total monthly hours per freelance worker – must be computed dynamically within ClickDown before being visualized in charts or tables.

Figure 4 illustrates a simplified data-flow model that connects the ClickUp API structure with ClickDown's internal logic. The system retrieves task-level data via API calls, processes it through a backend synchronization service, and stores it in a local database to improve response times and reduce redundant API queries. The intermediate layer enables the front-end dashboard to present near real-time summaries while minimizing delays caused by network limitations or rate-limit restrictions. Beyond webhook updates, the backend also performs scheduled reconciliation – a periodic full download of selected data – to ensure no changes are missed in cases where a webhook fails to fire or its payload cannot be processed correctly.

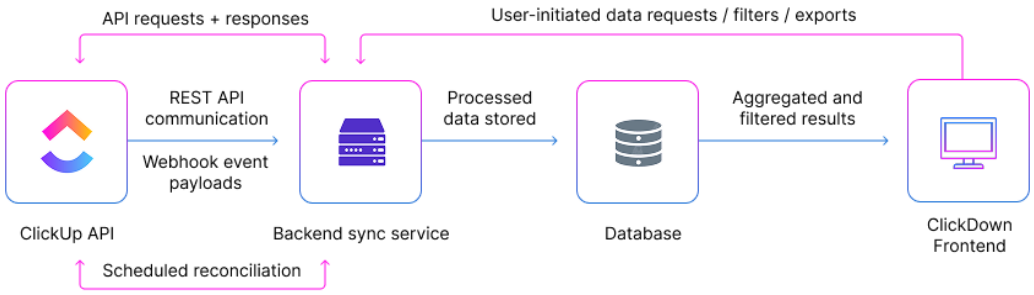

Figure 10: Simplified ClickDown data-flow model based on ClickUp API structure

Source: own elaboration based on ClickUp (2024c); ClickUp (2024f)

Real-time feedback in user interfaces depends on synchronization mechanisms that ensure consistency between what users see and what exists in the source system. To achieve outlined goals, ClickDown employs webhooks, which automatically transmit event updates (for example, a new time entry or task modification) from ClickUp to the ClickDown database (ClickUp, 2024i; ClickUp, 2024l). Each webhook delivers a JSON payload containing the relevant entity and change metadata, allowing the system to instantly update visual elements—such as recalculated totals or refreshed progress indicators—without manual page reloads. From a UX perspective,

the described mechanism supports Norman's (2013) principle of feedback, providing immediate confirmation that user actions or system events have been processed successfully.

The integration features also introduce specific design limitations. API rate limits constrain the frequency of real-time updates; date formats stored in UTC require conversion to user-specific time zones; and the high granularity of JSON responses can slow interface performance. Table 4 summarizes key technical constraints identified in the ClickUp API and their respective UX/UI implications for ClickDown.

Table 5: Technical constraints and UX/UI implications in ClickDown

Technical Constraint	***UX/UI Implication***	***Design Response***
API rate limits	Possible delay in data refresh	Introduce status indicators and manual refresh option
UTC timestamps	Incorrect local time display	Convert to local time zone automatically
Nested JSON data	Slower load time for reports	Use cached summaries and background pre-loading
Authentication tokens	Expired sessions disrupt continuity	Implement re-authentication prompts with minimal disruption
Limited endpoint fields	Missing metadata in views	Supplement with internal computed metrics

Source: own elaboration based on ClickUp (2024f)

The interplay between the constraints and user needs exemplifies the principle that "design is not free from the medium that supports it" (Norman, 2013). Rather than viewing technical limits as barriers, they serve as design drivers that help define interaction priorities and reduce complexity. In ClickDown, the considerations informed early decisions such as implementing cached monthly summaries, local timezone rendering, and a simplified export interface instead of continuous live querying.

The integration of ClickUp's data ecosystem into ClickDown demonstrates that usability and system architecture are interdependent. The success of the reporting tool depends not only on its visual design but also on how effectively it transforms structured API data into meaningful insights. By acknowledging data and integration constraints as inherent components of the design process, the solution aligns with ISO 9241-210's (2019) call for iterative, context-aware development, where both technical and human factors are equally considered. The perspective ensures that ClickDown remains reliable, responsive, and transparent, translating the complexity of ClickUp's back-end into an interface that supports clarity, trust, and decision-making in everyday workflows.

2 Practical part

The practical part of the thesis translates the theoretical foundations of user-centred design into a concrete design process for Allodium Games. The initial stakeholder analysis confirmed that core project coordination processes were already adequately supported by ClickUp. The real workflow friction emerged at a different point: at the intersection of time tracking and financial reporting. Freelance workers had no reliable way to verify their monthly hours, and the financial analyst depended on a manual, disconnected process to prepare invoices.

ClickDown was designed in direct response to the identified needs. Built as a dedicated reporting layer on top of ClickUp, it addresses the specific gaps in transparency and financial oversight without replacing the existing tooling. The following chapters document how the current workflow was analysed, how the requirements were derived from stakeholder consultations, and how the design process led to the final prototypes.

2.1 Current workflow and problem analysis

A detailed understanding of the existing workflow is essential for defining the requirements of the ClickDown reporting tool. ClickUp serves as the central platform for project coordination and time tracking at Allodium Games, yet its native features do not fully meet the internal reporting and invoicing needs of the company. Freelance workers use ClickUp to record their daily work on tasks; financial processing is carried out externally through exported files and additional scripts. The resulting fragmentation introduces inefficiencies, inconsistencies, and additional administrative overhead.

The chapter examines how time is currently tracked, reviewed, and processed, and identifies where the main difficulties arise. By analysing the perspectives of freelance workers, the CTO, and the financial analyst, the following sections highlight the structural gaps in the existing workflow and the problems that the new reporting interface aims to resolve.

2.1.1 Overview of current time tracking in ClickUp

The structure described below is based on direct observation of Allodium Games' ClickUp workspace conducted during the thesis research period.

Time tracking at Allodium Games is performed entirely within ClickUp using its built-in timer and manual entry functions. All work related to client projects is organised within a dedicated internal structure that follows ClickUp's hierarchical model. The company uses a Time Tracking space, inside which all projects are grouped into a Projects folder. Each project has its own list, and each project list contains tasks named according to the pattern "[client name] + project name". Freelance workers locate the relevant task for the client they are currently working on and record their hours directly on it.

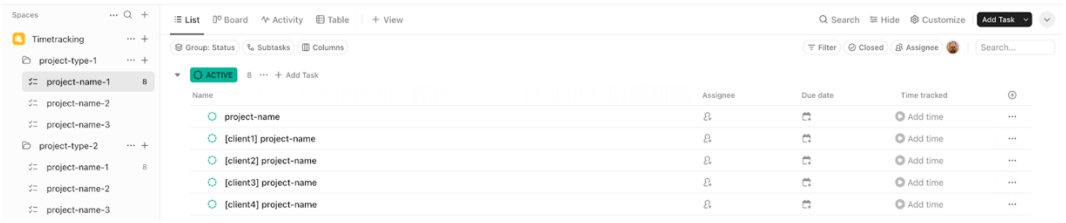

Figure 11: Internal structure used for time tracking in ClickUp

Source: own elaboration (2025)

Each time entry is stored as an individual record linked to a user and a task. ClickUp captures the duration and context of work, but the platform provides limited tools for freelance workers to review their total hours for longer periods. The only built-in feature available for personal review is the Timesheet view, which presents work hours grouped by week. Freelance workers who wish to check their monthly totals must therefore manually add weekly values or compare them with the monthly summary later exported by the CTO.

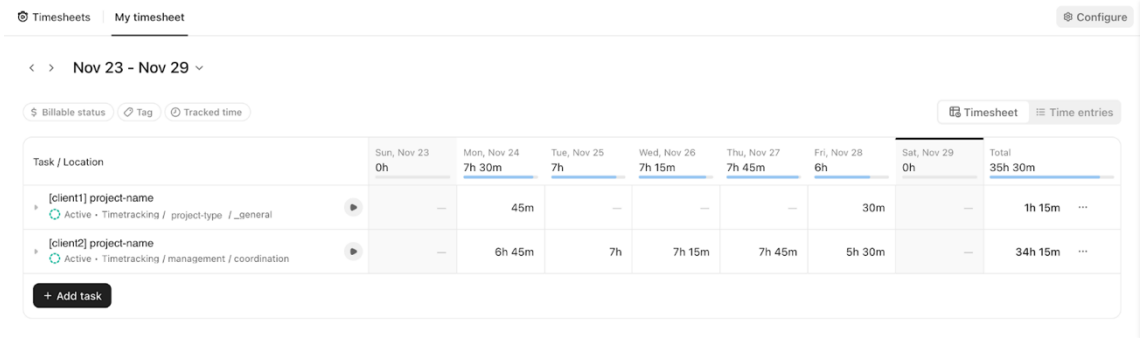

Figure 12: Weekly Timesheet display without a monthly overview

Source: own elaboration (2025)

ClickUp does not offer a per-user monthly total or a consolidated calendar-month view. Freelance workers rely on two methods to verify their hours: summing the weekly totals displayed in the Timesheet or cross-checking against the monthly CSV summary shared internally. The manual process is relatively error-prone and time-consuming, particularly at the end of the month when accuracy is required for invoicing.

From the perspective of the financial analyst, ClickUp does not provide any interface that aggregates hours by freelance worker or client. The platform's reporting tools are oriented toward project management rather than financial workflows, meaning that no native view offers the structure needed for payroll or client-based invoicing. Instead, all detailed monthly summaries must be generated manually through exports performed by the CTO and processed outside ClickUp.

The combination of the identified limitations results in a workflow where freelance workers have limited visibility into their own performance metrics, and the accountant lacks a reliable, ready-to-use reporting interface. The disconnect forms one of the main motivations for designing ClickDown: a unified and user-friendly system that transforms raw ClickUp time-tracking data into structured monthly reports suitable for both freelance workers and financial processing.

2.1.2 CTO Interview: technical constraints and opportunities

The consultation took the form of a semi-structured expert interview conducted via Zoom. Questions were prepared in advance and focused on three areas: the existing technical infrastructure for time-tracking data, the constraints of the ClickUp API, and the planned development environment for ClickDown. Responses were recorded in written form during the session.

The consultation with the CTO of Allodium Games provided essential clarification of how time-tracking data from ClickUp is currently processed and how the technical processes determine the possibilities and limitations of the ClickDown reporting tool. Freelance workers track their time directly inside ClickUp, yet the CTO explained that all reporting activities rely on an internal infrastructure rather than on live API calls. Data from ClickUp is retrieved through the REST API and synchronised into a local PostgreSQL database that mirrors the ClickUp hierarchy of spaces, folders, lists, tasks, and time entries. The copy serves as the operational source of truth, as direct querying of ClickUp would be too slow for everyday reporting.

To ensure that the internal dataset remains consistent, two mechanisms are used simultaneously. Webhooks deliver real-time notifications whenever a relevant entity changes; a scheduled reconciliation process periodically re-downloads selected data in bulk to avoid the risk of missed updates. Together, the mechanisms maintain an accurate reflection of ClickUp's state, which is essential for the reliability of ClickDown's interface.

The interview also clarified that several key limitations of the ClickUp API directly shape the design of the reporting tool. Time entries are returned only as individual records, meaning that all aggregations (such as monthly summaries per freelance worker or per client) must be computed internally. All timestamps are stored in UTC and need to be converted to local time during presentation. Important attributes required for financial reporting, such as hourly rate or type of work, are not available in ClickUp and must therefore be stored in the internal system. Authentication for ClickDown will rely on ClickUp's Single Sign-On and operate with read-only permissions, reinforcing the principle that ClickDown can display data but cannot modify ClickUp content.

An additional point discussed during the interview concerned client classification. Because ClickUp does not offer a dedicated field for client identification, the current workflow uses a script that scans task names and matches them against a predefined list of client names. The logic will remain important for ClickDown, since missing matches result in undefined client values that the accountant must later verify manually.

The CTO also outlined the technical environment in which ClickDown will be developed. The front-end will be built using Svelte, the back-end will run on Node.js with TypeScript, and PostgreSQL will serve as the database layer. Internal UI component libraries will be reused to maintain consistency. The system will rely primarily on existing ClickUp API endpoints, though custom internal endpoints may be introduced to optimise performance and reduce complexity on the client side.

The interview confirmed that the design of ClickDown must closely follow the structure and limitations of ClickUp's API and leverage the company's internal synchronisation layer. The insights define the realistic boundaries of the solution and ensure that the subsequent interface

design is aligned with the technical foundations of the system. Responses were recorded in written form during the session. A structured summary of the interview is provided in Appendix A.

2.1.3 Accountant workflow and pain points

The accountant's workflow was examined through a semi-structured interview conducted via Zoom. Questions were prepared in advance and focused on the monthly invoicing process, the data sources currently in use, and the main difficulties encountered during financial reporting. Responses were recorded in written form during the session.

The monthly invoicing process at Allodium Games relies heavily on the data exported from ClickUp, yet the workflow remains largely manual and time-consuming. After all freelance workers finish tracking their hours for the month, the CTO generates a CSV report by downloading raw time-tracking data from ClickUp and processing it through an internal script. The script consolidates the time entries into a simplified table containing five key columns: year, month, freelance worker name, client, and time in hours. The accountant receives the file as the basis for all subsequent financial operations.

Table 6: Example structure of a monthly hours report (illustrative)

year	***month***	***name***	***client***	***time in hours***
2025	11	Harry Bonbon	client3	31.04

Source: own work based on Allodium Games' internal workflow (2025)

The accountant begins each month by opening the aggregated CSV report and reviewing hour allocations by freelance worker and client. The most important dimension in the dataset is the client, because it directly determines how labour costs are allocated and later used in the company's cashflow calculations. Since ClickUp does not store the client as a structured field, the internal script must infer it by scanning task names and matching them against a predefined list of client identifiers. When the script finds a match, the time entry is assigned to that client. If no match is found, the client column remains empty ("–"), and the accountant must manually contact the relevant freelance worker for clarification. The mismatch introduces delays and increases the likelihood of inconsistencies, especially when tasks are named incorrectly or deviate from the expected naming convention.

The exported CSV consolidates the most essential information, yet it does not include several attributes needed for financial processing. Hourly rates and types of work (for example, development, design, coordination) are managed in separate internal tables outside ClickUp. During the invoicing stage, the accountant manually looks up the appropriate hourly rate for each freelance worker and calculates the monthly payout. Freelance workers in coordination, management, or administrative roles typically do not bill hours to clients; their internal workload may still need to be tracked for operational oversight, which the current CSV does not fully support.

Several pain points were identified in the workflow. First, the accountant lacks a dedicated interface that would present monthly hours in a clear, structured, and filterable way. The current CSV offers limited options for verification, summarisation, or grouping. Second, manual correction of missing or ambiguous client names is frequent and depends on freelance workers

providing additional explanations. Third, the process relies on multiple disconnected data sources: time entries from ClickUp, hourly rates from separate spreadsheets, and operator classifications from another system, which increases administrative overhead. Fourth, because freelance workers do not have an integrated monthly overview inside ClickUp, they are unable to easily validate their own hours before the accountant receives the final CSV, leading to avoidable errors and follow-up communication.

The identified issues highlight a broader systemic lack of transparency and automation in the reporting process and demonstrate the need for a specialised tool like ClickDown, which can unify the data, provide consistent client classification, and support both freelance worker self-verification and efficient financial processing. Responses were recorded in written form during the session. A structured summary of the interview is provided in Appendix B.

2.1.4 Key pain points identified

The analysis of the current time-tracking and reporting workflow reveals several interconnected pain points affecting freelance workers, the financial analyst, and the overall reliability of reporting outputs. ClickUp provides a functional mechanism for recording time at the task level, yet it lacks the aggregation and visibility required for monthly reporting and financial processing. Both operational and administrative activities therefore rely on manual effort and external tools.

From the freelance workers' perspective, the most significant limitation is the absence of a clear monthly overview of worked hours. ClickUp's Timesheet view displays time data only in weekly segments, forcing users to manually sum values to determine their total hours for a given month. The process is further complicated by the lack of any structured view showing how hours are distributed across clients. Freelance workers cannot independently verify whether their tracked time has been correctly attributed to the appropriate client or whether any task naming inconsistencies may affect downstream reporting. Workers therefore rely on a secondary CSV report provided at the end of each month to validate their hours, introducing redundancy and uncertainty into the workflow.

The financial analyst faces a separate but related set of challenges. Monthly reporting currently depends on a CSV file generated by a custom script that extracts time data from ClickUp and attempts to classify hours by client based on task name patterns. The approach enables basic aggregation, yet it is inherently fragile. If a task name does not contain a recognizable client identifier, the resulting report includes unclassified entries marked with missing client values. In such cases, the accountant must manually contact freelance workers to clarify the correct client, delaying invoice preparation and increasing administrative overhead. Additional risks arise from data handling itself, as numeric hour values may be incorrectly interpreted by spreadsheet software, requiring further manual correction.

A deeper structural issue lies in the separation of data sources used for financial calculations. Worked hours are obtained from ClickUp, yet information such as hourly rates and types of work is maintained in separate internal tables. The absence of a unified interface means that the accountant must manually combine multiple datasets to calculate payments and allocate costs. The fragmentation reduces transparency and increases the likelihood of errors, particularly as the volume of tracked time grows.

From a technical standpoint, several confirmed constraints directly influence the problems identified above. ClickUp's API exposes time entries only as individual records, requiring all aggregation logic to be implemented externally. All timestamps are stored in UTC, necessitating local time conversion during presentation. ClickUp does not store essential financial metadata such as hourly rates or work categories, making it impossible to derive complete reports without maintaining an internal database. Client classification currently depends on task naming conventions; the limitation cannot be eliminated in the short term due to platform constraints and must be mitigated through interface design rather than data restructuring.

The identified issues highlight a fundamental mismatch between how time is recorded in ClickUp and how it is consumed for reporting and financial decision-making. The absence of a centralized monthly overview, the reliance on fragile naming conventions, the lack of an accountant-facing reporting interface, and the separation of operational and financial data collectively define the core UX problem addressed in the present thesis. ClickDown is conceived not as a replacement for ClickUp's time tracking, but as a complementary reporting layer that transforms raw time data into structured, transparent, and verifiable information aligned with the needs of all stakeholders.

2.2 Requirements specification for ClickDown

The workflow analysis conducted in Section 2.1 established a clear picture of the operational and technical limitations affecting the current reporting process at Allodium Games. Section 2.2 translates the identified problems and stakeholder needs into a structured requirements specification for ClickDown. The section covers three areas: the identification of stakeholder groups and user roles, the definition of functional requirements describing what the system must do, and the specification of non-functional requirements describing the conditions under which the system must operate.

2.2.1 Stakeholders and user roles

Identifying stakeholders and user roles is a key step in the human-centred design process, as defined by ISO 9241-210. The standard emphasizes that system requirements must be derived from a clear understanding of users, their tasks, and the organizational context in which the system is used. In the case of ClickDown, an internal reporting application, the step is essential because the same dataset supports users with different responsibilities and decision-making needs.

The stakeholder analysis presented in the section is based on the workflow analysis described in Section 2.1 and on consultations with company representatives. In line with user-centred design methodology and Nielsen's usability engineering approach, stakeholders are not defined by technical permissions but by their goals and patterns of interaction with reporting data. The approach ensures that role definitions reflect real usage rather than assumed system functionality.

Four stakeholder groups were identified as relevant to the ClickDown system: freelance workers, the accountant, management, and the CTO as a technical stakeholder. All groups rely on data

originating from ClickUp, yet their interaction with ClickDown differs in purpose, frequency, and level of detail.

Freelance workers represent the largest group of end users. Their primary responsibility is to track time on tasks directly in ClickUp using the built-in timer or manual entries. ClickDown does not replace the process but provides a complementary reporting layer that allows freelance workers to review how their tracked time is aggregated at the monthly level. As shown in Section 2.1, ClickUp does not offer a native monthly overview per freelance worker, which forces users to manually calculate totals or rely on external CSV reports.

Within ClickDown, freelance workers use the system mainly to verify the correctness and completeness of their reported hours before the data is processed for financial purposes. Their interaction is occasional and typically occurs near the end of a reporting period. Freelance workers do not require advanced filtering or analytical tools. They depend on clear, accurate, and transparent presentation of aggregated data. Freelance workers are therefore characterized as users with a verification-oriented role, focused on understanding and confirming reported information rather than manipulating it.

The accountant represents the most analytically demanding user group. The role is responsible for preparing invoices, allocating labour costs to clients, and supporting cashflow calculations. Unlike freelance workers, the accountant works with time-reporting data on a regular basis and combines it with additional financial information, such as hourly rates and work classifications, which are stored outside ClickUp.

As described in Section 2.1.3, the current workflow relies on CSV exports generated through a custom script. The files provide only limited opportunities for validation, filtering, and summarization, which increases administrative effort and the risk of errors. In ClickDown, the accountant's role is therefore defined by the need to review and interpret aggregated data across multiple dimensions, including freelance workers, clients, and reporting periods. Because the system operates in read-only mode, the accountant's interaction is limited to exploration, validation, and export of data. The accountant is the primary analytical user whose needs strongly influence reporting structure and information hierarchy.

Management forms a stakeholder group with a strategic rather than operational interest in reporting data. Management does not participate directly in time tracking or invoicing processes. Instead, reporting outputs are used to gain a high-level overview of workload distribution and client effort over time. Interaction with ClickDown is therefore infrequent and overview oriented.

For management users, the system should support fast interpretation of summarized information without exposing unnecessary detail. Consistent with usability principles that emphasize simplicity and minimalism, management access is intentionally limited to aggregated views. Management is classified as an overview-oriented user, whose requirements are secondary to those of freelance workers and the accountant.

The CTO is included as a technical stakeholder rather than an interface user. The role defines the technical boundaries of the system, including data synchronization, aggregation logic, authentication, and compliance with ClickUp API limitations. The CTO does not interact directly with ClickDown's interface, yet the constraints have a direct impact on what data can be

presented and how it can be structured. Explicitly acknowledging the role ensures that design decisions remain technically feasible and aligned with the existing infrastructure.

Table 7 summarizes the identified stakeholder groups and their relationship to the ClickDown system.

Table 7: Overview of stakeholder roles and interaction goals in ClickDown

Stakeholder	***Primary goal***	***Interaction frequency***	***Role classification***
Freelance workers	Verify monthly worked hours and client distribution	Occasional, period-based	Verification-oriented user
Accountant	Prepare invoices and validate client-based hour allocation	Regular, systematic	Primary analytical user
Management	Monitor workload and client effort at an aggregated level	Infrequent	Overview-oriented user
CTO	Define technical constraints and data availability	Indirect	Technical stakeholder

Source: own work (2025)

ClickDown supports a heterogeneous group of stakeholders whose needs differ in scope and complexity. By defining user roles based on observed workflows and validated responsibilities, the stakeholder analysis establishes a solid foundation for the requirements specification presented in the following section. The differentiation of roles ensures that subsequent design decisions remain aligned with human-centred design principles and address real, documented needs rather than hypothetical use cases.

2.2.2 Functional requirements

Functional requirements describe the essential behaviour of the ClickDown system and define what the application must provide in order to support its intended users. In accordance with human-centred design principles, the requirements are derived directly from the workflow analysis presented in Section 2.1 and from the stakeholder roles identified in Section 2.2.1. The purpose of the section is to formalize system functionality without prescribing specific interface solutions or technical implementations.

ClickDown is designed as a read-only reporting layer that builds on time-tracking data recorded in ClickUp. The system does not support the creation, modification, or deletion of time entries, tasks, or financial data. Its functional scope is therefore intentionally limited to the aggregation, presentation, filtering, and export of existing data. The limitation reflects both the technical constraints of the ClickUp API and the organizational decision to preserve ClickUp as the single source of truth for time tracking.

One of the core functional requirements of ClickDown is the ability to present aggregated monthly overviews of worked hours. Freelance workers must be able to view their total number of worked hours for a selected calendar month, calculated automatically from individual time entries synchronized from ClickUp. In addition to the overall total, the system must provide a breakdown of worked hours by client. The requirement addresses the lack of monthly and client-based visibility in ClickUp and enables freelance workers to independently verify how their time is allocated before it is processed for financial reporting.

To ensure clarity and data protection, the system must restrict freelance workers to viewing only their own time data. The restriction prevents unnecessary exposure of information about other freelance workers and supports focused, individual verification. The system must also clearly indicate time entries that could not be assigned to a client through automated detection logic. Such entries represent a known weakness in the current workflow and must be visible to users so that potential issues can be identified early and resolved through external communication if necessary.

ClickDown must also support the needs of financial processing. The system must allow aggregated monthly reports to be viewed across all freelance workers and clients, enabling a comprehensive overview of time allocation for invoicing purposes. The reports must support filtering by reporting period, freelance worker, and client, allowing targeted inspection of specific subsets of data. The system must also provide aggregated summaries that can be used directly as input for financial calculations, such as total hours per client or per freelance worker within a given month.

A further functional requirement concerns data export. The system must allow reporting outputs to be exported in a structured format that is compatible with downstream invoicing workflows. Exported data must reflect exactly the values presented in the interface in order to maintain consistency and trust in the reporting process. Because ClickDown operates in read-only mode, no exported data can be modified within the system itself.

For management users, the system must provide access to high-level, aggregated monthly overviews that support strategic orientation rather than operational processing. The overviews must present workload distribution across clients and freelance workers in a standardized form that enables comparison between reporting periods. Management access must exclude detailed task-level or entry-level views in order to preserve simplicity and prevent unnecessary cognitive load.

Several functional requirements apply to all users regardless of role. The system must authenticate users via ClickUp Single Sign-On and restrict access based on user identity. All presented data must be derived from synchronized ClickUp records and aggregated internally, as ClickUp does not provide server-side aggregation. Reporting periods must be selectable using calendar months, and all timestamps must be converted from UTC to local time zones before presentation. The requirements ensure consistency, accuracy, and usability across all reporting views.

Table 8 summarizes the key functional requirements of the ClickDown system and indicates their primary stakeholders.

Table 8: Functional requirements of the ClickDown system

ID	*Requirement description*	*Primary stakeholder*
FR1	The system shall display total worked hours per freelance worker for a selected calendar month.	Freelance workers
FR2	The system shall display worked hours grouped by client for a selected month.	Freelance workers
FR3	The system shall restrict freelance workers to viewing only their own time data.	Freelance workers
FR4	The system shall indicate time entries with missing or undefined client assignment.	Freelance workers, Accountant
FR5	The system shall provide monthly time reports across all freelance workers and clients.	Accountant
FR6	The system shall support filtering of reports by month, freelance worker, and client.	Accountant
FR7	The system shall provide aggregated summaries suitable for invoicing preparation.	Accountant
FR8	The system shall allow export of reporting data in a structured format.	Freelance workers, Accountant
FR9	The system shall provide high-level aggregated monthly overviews without task-level detail.	Management
FR10	The system shall authenticate users via ClickUp SSO and operate in read-only mode.	All users

Source: own work (2025)

The functional requirements defined in the section translate the identified workflow problems and stakeholder needs into a coherent specification of system behaviour. Rather than introducing new processes or altering existing time-tracking practices, ClickDown focuses on transforming already available data into structured, transparent, and verifiable reporting outputs. The requirements deliberately reflect the read-only nature of the system and the technical limitations of the ClickUp API, ensuring that proposed functionality remains realistic and aligned with the existing infrastructure.

By grounding functional requirements in observed user roles and validated organizational needs, the specification establishes a stable foundation for subsequent design decisions. Functional completeness alone is insufficient to guarantee usability and system acceptance. Aspects such as performance, reliability, data freshness, and interaction constraints play an equally important role in shaping user experience. The considerations are addressed in the following section, which defines the non-functional requirements that further frame the design and feasibility of the ClickDown reporting tool.

2.2.3 Non-functional requirements

Functional requirements define what the ClickDown system must do; non-functional requirements specify the conditions under which the functionality must operate. Non-functional

requirements address qualities such as performance, reliability, security, and usability, which significantly influence user experience but are not tied to individual features. In a reporting-oriented system such as ClickDown, non-functional aspects are particularly important because users depend on the system's responsiveness, accuracy, and consistency when validating time data and preparing financial outputs.

The non-functional requirements presented in the section are derived from two main sources. The first is the workflow analysis described in Section 2.1, which revealed performance bottlenecks, risks related to data consistency, and usability issues caused by delayed or fragmented reporting. The second source consists of confirmed technical constraints discussed with the CTO, including API rate limits, asynchronous data synchronization, and the use of a local database as an intermediary layer. The factors define the quality attributes that the system must satisfy in order to be accepted by its users.

A key non-functional requirement concerns system performance. ClickDown must provide reporting views with acceptable response times even when processing large volumes of time-tracking data. Because the ClickUp API does not support server-side aggregation and is subject to rate limits, the system cannot rely on real-time API calls for every user interaction. Aggregated data must therefore be computed internally and retrieved from a local database. From a user experience perspective, reporting views must load quickly enough to support exploratory use, particularly for the accountant, who interacts with the system regularly. Where immediate freshness of data cannot be guaranteed, the system must communicate the limitation clearly to avoid misinterpretation.

Closely related to performance is the requirement for data freshness and consistency. ClickDown must present data that reflects the current state of time entries in ClickUp with a reasonable delay. Because synchronization is performed through a combination of webhooks and scheduled reconciliation jobs, temporary discrepancies may occur. The system must therefore ensure internal consistency of displayed values and avoid presenting partially updated or contradictory information. From a usability standpoint, it is essential that users understand whether they are viewing the most recent data or a cached summary.

Reliability represents another critical quality attribute. The system must handle synchronization failures, missing data, or incomplete records in a predictable and transparent manner. Rather than failing silently, ClickDown must surface anomalies such as missing client assignments or delayed updates in a way that supports user trust. The requirement is particularly important for the accountant, whose work depends on the correctness and completeness of monthly summaries. Reliable behaviour also includes consistent calculation logic, ensuring that the same input data always produces the same aggregated results.

Security and access control form a further category of non-functional requirements. Because ClickDown processes internal company data related to working hours and client allocation, access must be restricted to authenticated users. Authentication must be performed via ClickUp Single Sign-On, and authorization must ensure that users can only access data relevant to their role. Freelance workers must be limited to their own time data; broader access is reserved for authorized roles. The system must also operate strictly in read-only mode, preventing any modification of source data and reducing the risk of accidental or unauthorized changes.

Usability-related non-functional requirements address how easily users can understand and operate the system. Reporting views must be designed to minimize cognitive load and support quick interpretation of aggregated data. The requirement includes consistent interaction patterns, predictable navigation, and clear visual hierarchy. Because ClickDown is used by non-technical users, the interface must avoid exposing technical details related to data synchronization, API limitations, or internal processing logic. The constraints must be handled implicitly or communicated through simple, non-intrusive feedback.

Maintainability and scalability must also be considered, even though ClickDown is an internal tool. The system must allow for future adjustments to reporting logic, such as the introduction of new clients or changes in internal classification rules, without requiring fundamental redesign. Large-scale growth is not the primary goal, yet the architecture and design must remain adaptable to incremental changes in organizational needs.

Table 9 summarizes the key non-functional requirements and their implications for the ClickDown system.

Table 9: Non-functional requirements of the ClickDown system

Category	***Requirement description***	***Design implication***
Performance	Reporting views shall load within acceptable response times using internally aggregated data.	Use cached summaries and local database queries
Data freshness	Displayed data shall reflect synchronized ClickUp records with clearly defined update latency.	Communicate update state and avoid silent delays
Reliability	Aggregations shall be consistent and anomalies shall be clearly indicated.	Predictable calculations and visible error states
Security	Access shall be restricted via ClickUp SSO and role-based authorization.	Enforced read-only access and identity-based views
Usability	The system shall support clear, low-effort interpretation of reports.	Consistent layouts and minimal cognitive load
Maintainability	Reporting logic shall allow incremental changes without redesign.	Modular structure and configurable rules

Source: own work (2025)

The non-functional requirements defined in the section establish the quality attributes that enable ClickDown to function as a trustworthy and usable reporting tool. By addressing performance, data consistency, reliability, security, and usability alongside technical constraints, the requirements ensure that the system supports real-world workflows rather than idealized scenarios. User experience in reporting systems depends not only on visible features but on the stability and transparency of underlying processes.

The functional and non-functional requirements defined in Sections 2.2.2 and 2.2.3 form a comprehensive foundation for the subsequent design phase. The following chapter builds on the requirements to describe the proposed design and interaction structure of ClickDown, showing how abstract requirements are translated into concrete reporting views and user flows.

2.3 UX/UI design process

The chapter describes the UX/UI design process applied during the development of the ClickDown reporting tool. The process follows a user-centred and constraint-driven approach, reflecting the practical context of an internal reporting system integrated with an existing project management platform. Rather than focusing on visual styling at an early stage, the design process prioritizes understanding user roles, workflows, and data structures in order to ensure clarity, reliability, and alignment with real organisational needs.

The design activities presented in the chapter build directly on the problem analysis and requirements specification discussed in the previous sections. Particular emphasis is placed on information architecture and interaction logic, as the aspects are critical for reporting tools where correctness, transparency, and trust in data are more important than visual expressiveness. The chapter documents the key design decisions, their rationale, and the resulting structure of the application.

2.3.1 Design process overview

The UX/UI design of ClickDown was conducted as a structured, step-by-step process that progressed from problem understanding toward concrete interface definition. The overall sequence of design activities is illustrated in Figure 13, which outlines the main phases applied during the project, from initial analysis to the creation of low-fidelity interface representations.

As shown in Figure 13, the process begins with problem analysis, focusing on the identification of user roles, existing workflows, and key pain points related to time tracking and reporting. The phase established the practical context of the system and defined the constraints that influenced all subsequent design decisions. Rather than approaching the interface from a visual perspective, the design intentionally started with an understanding of how reporting is performed within the organization and where inefficiencies occur.

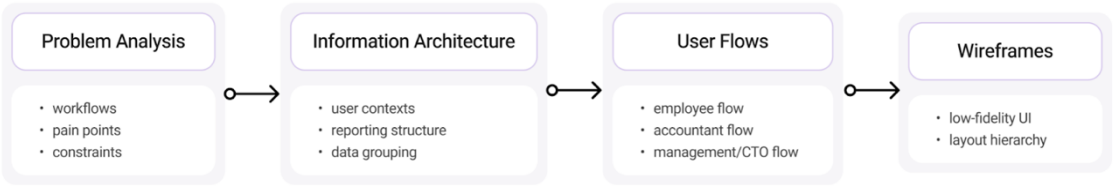

Figure 13: Overview of the UX/UI design process for ClickDown
Source: own work (2025)

The second phase of the process concentrates on information architecture. At the stage, the emphasis shifts from understanding the problem to structuring information in a way that reflects user responsibilities and reporting logic. The system was divided into role-based contexts, and the relationships between reporting periods, aggregated data, and export actions were defined. The step served as the foundation for the interface design, ensuring that later layout and interaction choices would be grounded in a clear and coherent structure.

Following the definition of information architecture, the design process continues with the creation of user flows. The flows describe how users move through the system when performing core tasks, such as verifying monthly worked hours or preparing aggregated reports for

invoicing. The purpose of the phase is to validate that the proposed structure supports real user actions without unnecessary complexity.

The final phase presented in Figure 13 involves the development of low-fidelity wireframes. The wireframes translate the previously defined structure and flow into spatial layouts, allowing verification of hierarchy, readability, and interaction order before introducing visual styling. High-fidelity design and visual refinement are deliberately postponed ensuring that usability issues are addressed at a structural level rather than being obscured by aesthetic elements.

The sequential approach supports clarity, reduces design risk, and enables transparent documentation of design decisions throughout the project.

2.3.2 Information architecture

Information architecture defines how information and functionality are structured within the application so that users can easily understand the system and complete their tasks efficiently. In the case of ClickDown, the information architecture was designed with a strong emphasis on role-based access, clarity of reporting data, and minimal cognitive load.

The overall structure of the application and its main reporting areas are illustrated in Figure 14.

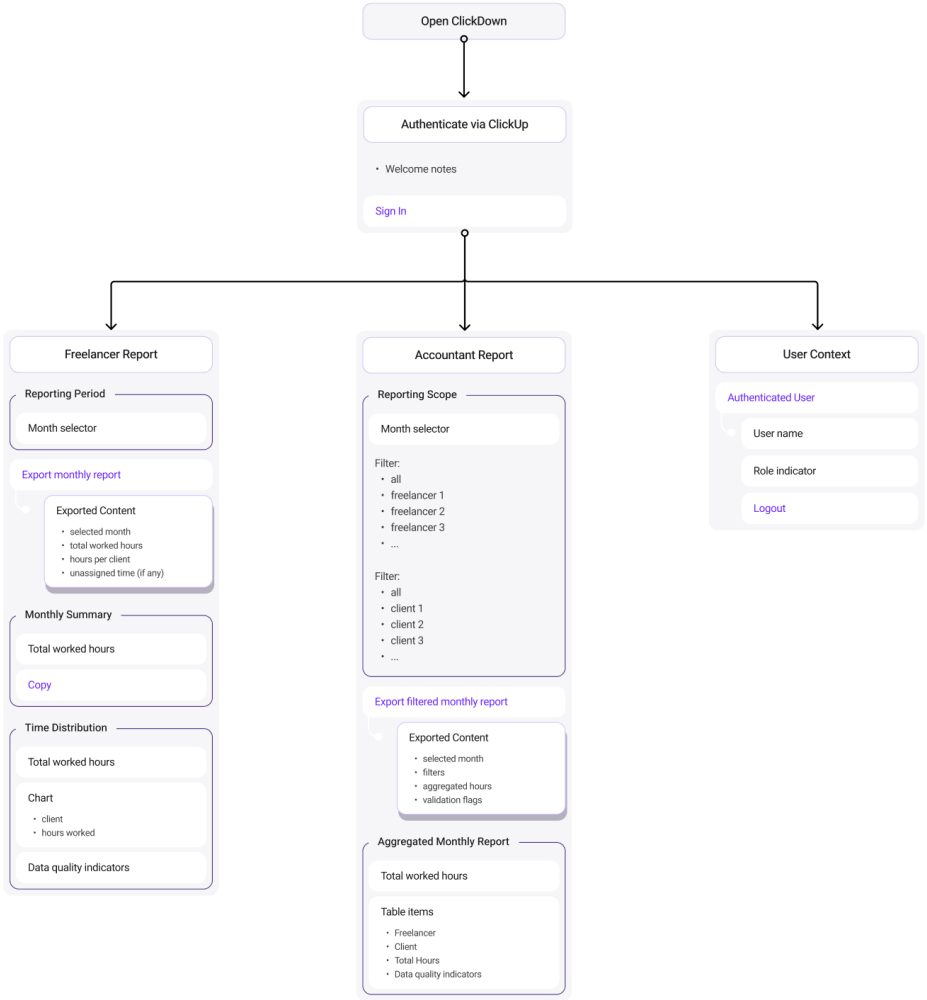

Figure 14: Information architecture of the ClickDown application

Source: own work (2025)

The starting point of the application is authentication via the ClickUp platform. The step ensures that only authorized users can access the system and that their role (freelance worker or accountant) can be identified immediately after login. Based on the authenticated role, users are directed to the relevant reporting view without unnecessary intermediate screens.

The application is structured around three primary areas: Freelance Worker Report, Accountant Report, and User Context. The areas represent the core user needs identified during the analysis phase and reflect the different perspectives from which reporting data is accessed.

The Freelance Worker Report focuses on individual time tracking and reporting. Its internal structure is divided into clearly separated sections, including the reporting period selection, monthly summary, time distribution, and export functionality. The structure allows freelance workers to quickly review their worked hours, understand how time is distributed across clients, and export reports when required.

The Accountant Report provides an aggregated view of time data across multiple freelance workers and clients. Instead of duplicating interface screens, the report extends the same architectural logic by introducing broader filters and validation indicators. The approach ensures consistency and supports more complex reporting and control tasks.

The User Context section contains user-specific information such as authentication status, username, role indicator, and logout functionality. Keeping the information separate from reporting views helps maintain clarity and prevents overloading analytical screens with non-essential elements.

Within the information architecture diagrams, different interface elements are visually distinguished to improve readability and interpretation. Sections represent logical groupings of content, elements indicate data blocks or fields, and actions such as buttons are highlighted using distinct visual styling. The notation supports both design planning and communication with stakeholders by making functional roles of interface components immediately understandable.

2.3.3 User flows

User flows were created to describe how different user types interact with the ClickDown application in order to achieve their goals. The flows are derived directly from the defined information architecture and reflect both functional requirements and access constraints based on user permissions. The primary purpose of the user flows is to validate that the proposed screen structure supports clear, efficient, and predictable user interaction without unnecessary steps.

The section focuses on the key interaction scenarios identified as essential for the application: authentication and access resolution, freelance worker reporting, accountant reporting, and management access. Each flow represents a sequence of actions and system responses rather than a visual layout, serving as a conceptual bridge between information architecture and wireframe design.

The first and most critical user flow addresses authentication and authorization. ClickDown does not manage user accounts independently; instead, it relies on ClickUp as an external identity

provider. The decision reduces redundancy, simplifies user management, and ensures consistency with existing organizational structures.

After opening the ClickDown application, the user is redirected to authenticate via ClickUp. Upon successful authentication, the system retrieves the user's workspace membership information from ClickUp, including the assigned role (Member, Admin, or Owner). The role is then used to determine which reporting views are available to the user.

Rather than introducing a separate role system inside ClickDown, the application reuses ClickUp membership levels as an authorization mechanism. Elevated roles (Admin or Owner) are interpreted as users with broader organizational responsibility, such as management or accounting; standard Members represent regular freelance workers.

Based on the evaluation, the system resolves access as follows:

- Users with Admin or Owner membership are granted access to both the Freelance Worker Report and the Accountant Report.
- Users with Member membership are granted access only to the Freelance Worker Report corresponding to their own logged time data.

The flow ensures that access control remains transparent and aligned with existing workspace governance, avoiding additional configuration overhead within the application itself.

Figure 15 illustrates the authentication and access resolution user flow, including the decision point where the user's ClickUp role determines the available reporting views.

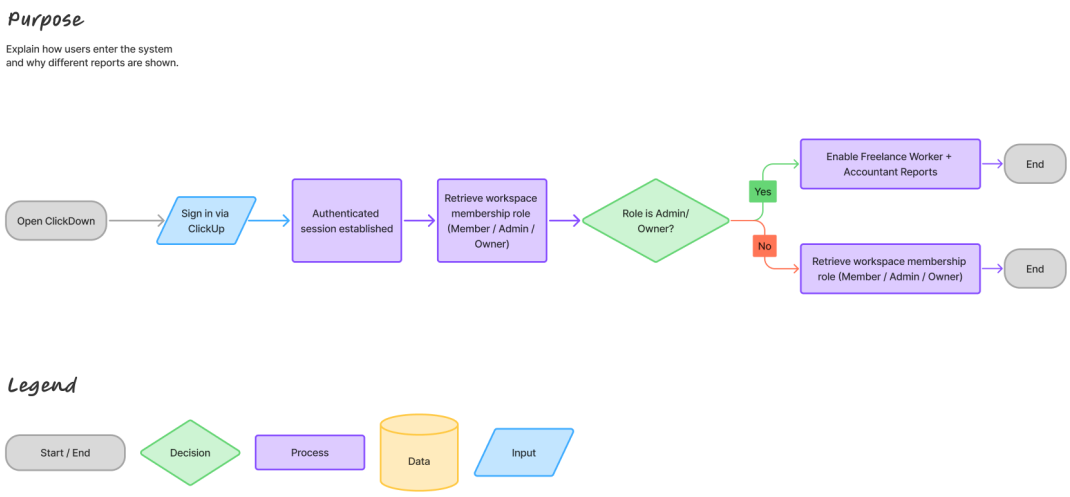

Figure 15: User Flow A – Authentication and Access Resolution
Source: own work (2025)

The freelance worker reporting flow describes how a standard workspace member interacts with ClickDown to review and verify personal working time for a selected month. The flow is designed to support transparency and self-verification while maintaining a read-only interaction model.

After authentication, the user is directed to the Freelance Worker Report view. The primary interaction within the view is the selection of a reporting period using a month selector.

Changing the selected month triggers reloading of all dependent data, including the total worked hours and time distribution.

Once the data is loaded, the user is presented with a summary of total worked hours and a visual distribution of time by operator. Due to inconsistent task naming in the source system, sometime entries cannot be reliably attributed to a specific operator. The entries are aggregated into an "Unassigned / Other" category within the distribution view.

To maintain clarity without hiding information, the "Unassigned / Other" category supports optional drill-down interaction. When expanded, the user is shown a grouped list of tasks for the selected period, where tasks with identical names are consolidated and their logged hours are summed. The approach avoids overwhelming the user with raw time logs and still enables verification of how unassigned time was generated.

The export action is available directly within the reporting period control area. Placing the export option next to the month selector reinforces the contextual relationship between the selected period and the generated output. When triggered, the export includes the selected month, total worked hours, time distribution by operator, and aggregated unassigned task data if present.

The flow intentionally excludes editing or correction of time entries, as ClickDown functions as a reporting and verification tool rather than a time-tracking system. The flow concludes either with on-screen review or by exporting the monthly report.

Figure 16 illustrates the freelance worker monthly reporting flow, including optional inspection of unassigned time and contextual export.

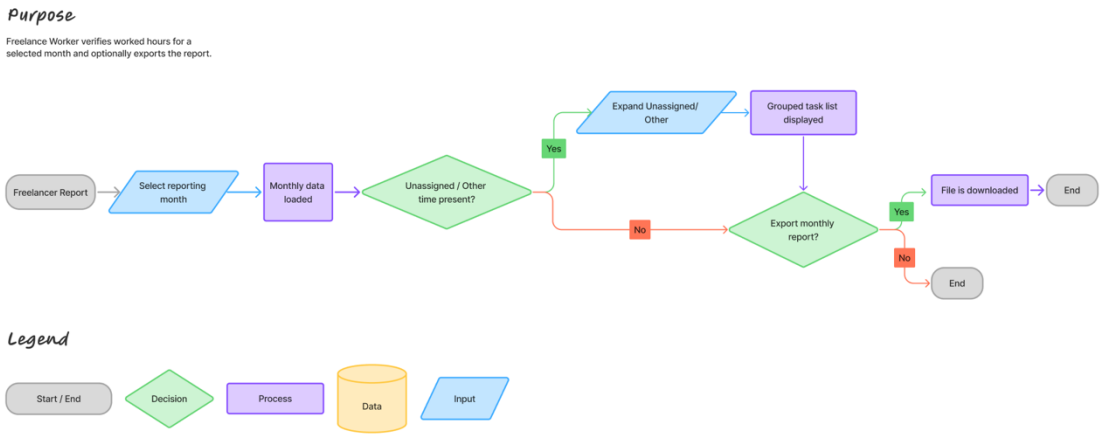

Figure 16: User Flow B – Freelance Worker Monthly Reporting

Source: own work (2025)

The accountant reporting flow describes how users with elevated workspace permissions (ClickUp Admin or Owner) interact with ClickDown to review, validate, and export aggregated monthly working time data across freelance workers and clients. The report is referred to as the Accountant Report, yet the designation reflects its primary use case (financial oversight, payroll preparation, and management reporting) rather than a distinct user role within the system.

After authentication and access resolution, users with Admin or Owner membership are granted access to both the Freelance Worker Report and the Accountant Report views. Upon opening the Accountant Report, the system automatically loads data for the current reporting period.

The initial state presents a baseline aggregated overview that includes all freelance workers and all clients, ensuring that meaningful information is available immediately without requiring preliminary interaction.

Figure 17 illustrates the aggregated monthly reporting user flow, showing the automatic loading of current data, subsequent refinement through filtering, handling of unassigned records, and the export action.

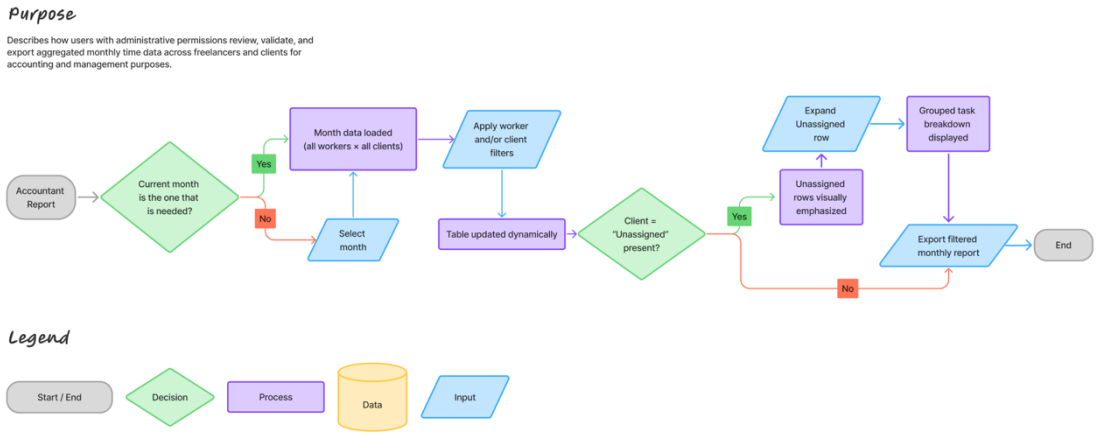

Figure 17: User Flow C – Accountant Aggregated Monthly Reporting

Source: own work (2025)

The primary interaction within the flow is the selection of the reporting period. If the current month does not correspond to the user's needs, a different month can be selected using the month selector, which triggers reloading of all dependent data. The aggregated table is updated dynamically to reflect the selected period.

The main reporting table summarizes working time grouped by freelance worker and client. By default, the table displays records for all freelance workers and all clients. Users may refine the view by applying freelance worker and/or client filters to inspect specific subsets of the data, such as analysing how a particular freelance worker contributed to a given client or reviewing all recorded work for a specific client across the organization.

Due to inconsistencies in task naming within the source system, sometime entries cannot be reliably attributed to a specific client. The records are aggregated under an "Unassigned" value in the Client column and are visually emphasized to support data quality validation. As shown in Figure 11, unassigned rows support optional drill-down interaction. When expanded, a grouped task breakdown is displayed, where tasks with identical names are consolidated and their logged hours are summed. The approach provides transparency while avoiding the exposure of raw time logs.

Once the desired data view is achieved, users may export the filtered monthly report. The exported output reflects the selected reporting period, applied filters, aggregated working hours per freelance worker and client, and any unassigned entries present in the dataset. The flow supports both accounting and managerial use cases through a single, unified reporting interface.

2.3.4 Low-fidelity wireframes

After completing the analytical phase – including stakeholder interviews, workflow mapping, and identification of technical constraints related to the ClickUp API – low-fidelity wireframes were developed. Design decisions were informed not only by user requirements but also by integration constraints described in Section X. Limitations related to API rate limits, non-real-time synchronization, UTC timestamp handling, and dynamically computed aggregates directly influenced interface structure and feedback mechanisms.

The objective of the wireframing stage was therefore not visual refinement, but validation of information hierarchy, structural relationships between components, and interaction logic across user roles under the technical conditions outlined above.

The wireframes were created for both desktop and mobile resolutions to ensure responsive behaviour of the system. During wireframing, grayscale layout and simplified elements were intentionally used to eliminate visual bias and concentrate on usability structure. The design decisions presented in Section 2.3.4 are grounded in user-centred design principles such as visibility of system status, cognitive load reduction, progressive disclosure, and structured data visualization (Norman, 2013; Nielsen, 1994; Few, 2013).

The authentication interface is presented in Figure 18 (desktop and mobile version).

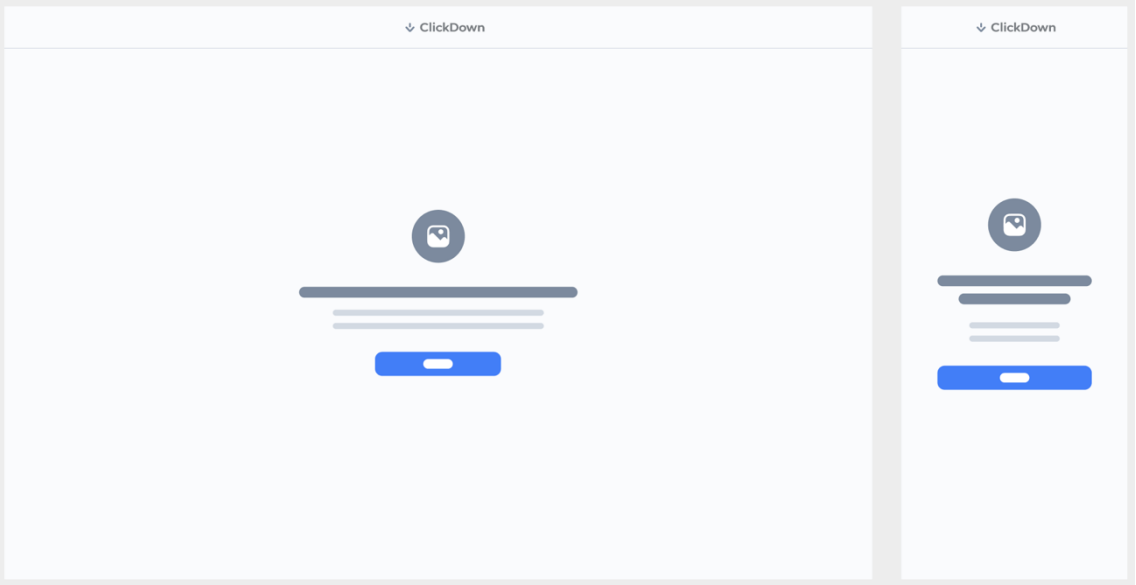

Figure 18: Authentication Screen (Desktop and Mobile)

Source: own work (2025)

The screen appears when the user accesses ClickDown for the first time. Authentication is handled through ClickUp Single Sign-On (SSO); therefore, the interface does not include additional credential fields. The layout contains only a welcome message and a primary action button that redirects the user to ClickUp authentication.

The minimal structure follows the principle of aesthetic and minimalist design (Nielsen, 1994). Since authentication logic is externally managed, the design focuses solely on clarity of entry and seamless transition into the system environment.

The Worker View in desktop and mobile resolution is shown in Figure 19.

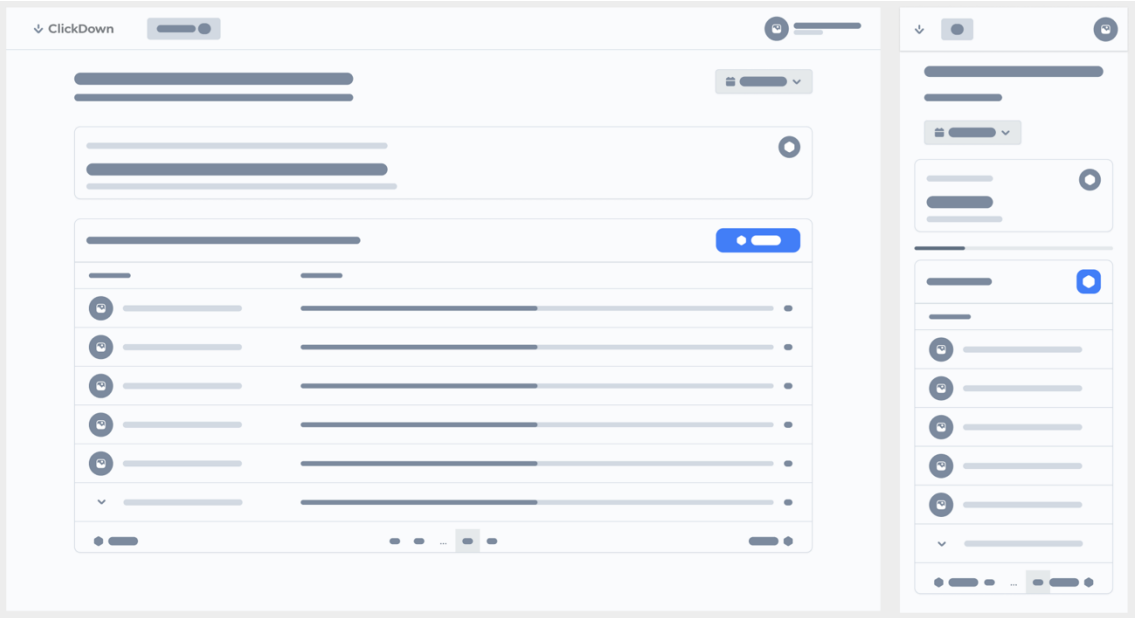

Figure 19: Worker View (Desktop and Mobile)

Source: own work (2025)

The primary purpose of the view is to provide freelance workers with a structured monthly overview of worked hours categorized by client. The layout follows a vertical information hierarchy consisting of:

- navigation,
- contextual report information,
- total hours summary,
- detailed client-level data.

The structure reflects the principle of "overview first, details on demand" (Few, 2013), ensuring that users first understand the scope of the report before interacting with detailed records.

The navigation includes the application logo, an active tab indicator, and the user profile section. The Worker View currently represents a single reporting context; the tab structure anticipates the accountant role, supporting scalability and consistency across roles. The profile area allows logout functionality, reinforcing visibility of system state (Norman, 2013).

The contextual section displays the report type, selected month and year, and "Last updated" information. The month selector is positioned in the contextual section because it affects the entire dataset displayed on the page. The placement establishes a clear mapping between control and system outcome (Norman, 2013).

The inclusion of the "Last updated" timestamp directly reflects synchronization constraints identified in Section 2.1.2. Since data retrieval depends on API calls, webhooks, and scheduled reconciliation rather than continuous live streaming, explicit visibility of update status reduces uncertainty and increases user trust in the accuracy of displayed information.

Below the contextual section, a summary card presents total worked hours in the format "000h 00m." The format was selected to enhance readability compared to decimal hour formats. The summary card includes a copy function, introduced based on workflow analysis. Freelance workers may reuse total hours when communicating with accounting or preparing invoice

documentation. Providing a direct copy action reduces repetitive manual tasks and supports efficiency of use (Nielsen, 1994). As individual time entries are retrieved as atomic records via the API, monthly totals must be computed dynamically within ClickDown before visualization. The summary card therefore represents an internally aggregated value rather than a directly stored field in the source system.

The main data representation consists of a table structured into two columns: Client and Hours Worked. Each row contains a client identifier (including logo where available), a horizontal progress bar, and the exact hour value. The progress bar communicates proportional distribution of time relative to total monthly hours, while the numeric value preserves precision. Combining visual and numeric encoding enhances interpretability (Few, 2013).

Pagination is included at the bottom of the table to support scalability in the event of increased client volume. The export function is placed in direct relation to the table to indicate that exported data corresponds to the currently displayed dataset.

The interaction of the month selector is illustrated in Figure 20.

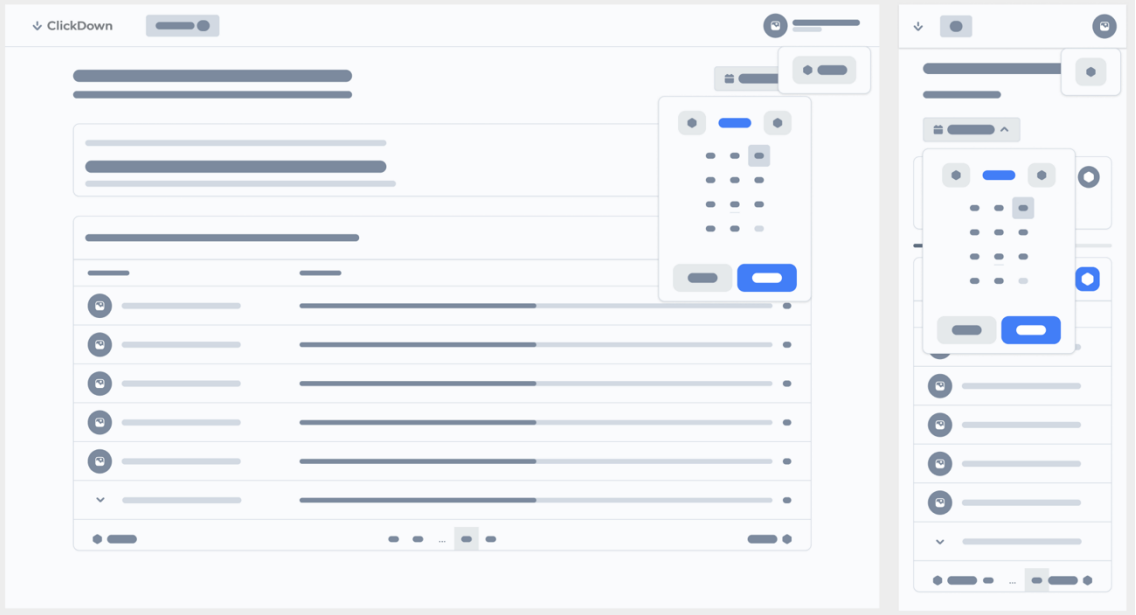

Figure 20: Worker View with Month Selector and Log Out Open (Desktop and Mobile)
Source: own work (2025)

The selection mechanism allows only month and year input. Daily or custom range selection was intentionally excluded because reporting requirements identified during research are strictly monthly. Limiting the scope of selection reduces interaction complexity and supports the principle of constraints in interaction design (Norman, 2013). Restricting selection to monthly intervals also aligns with the internal caching and aggregation strategy, which is optimised for month-level summaries rather than arbitrary date ranges.

The expanded state of unassigned records is presented in Figure 21.

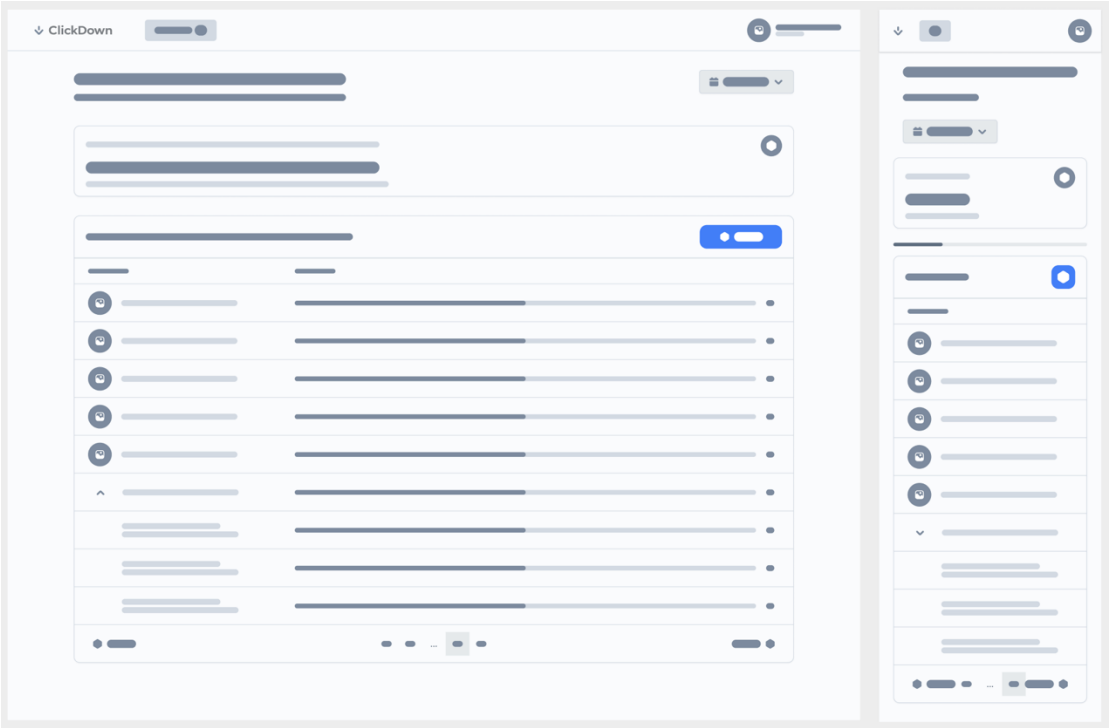

Figure 21: Worker View with Unassigned Section Expanded (Desktop and Mobile)

Source: own work (2025)

Tasks without associated client information are grouped under an "Unassigned" category. The row can be expanded to reveal task-level detail. The expandable mechanism applies progressive disclosure, preventing information overload while maintaining transparency. The feature supports communication between freelance workers and accountants in cases where time categorization requires clarification.

The Accountant View in collapsed state is shown in Figure 22, and the expanded state is shown in Figure 23.

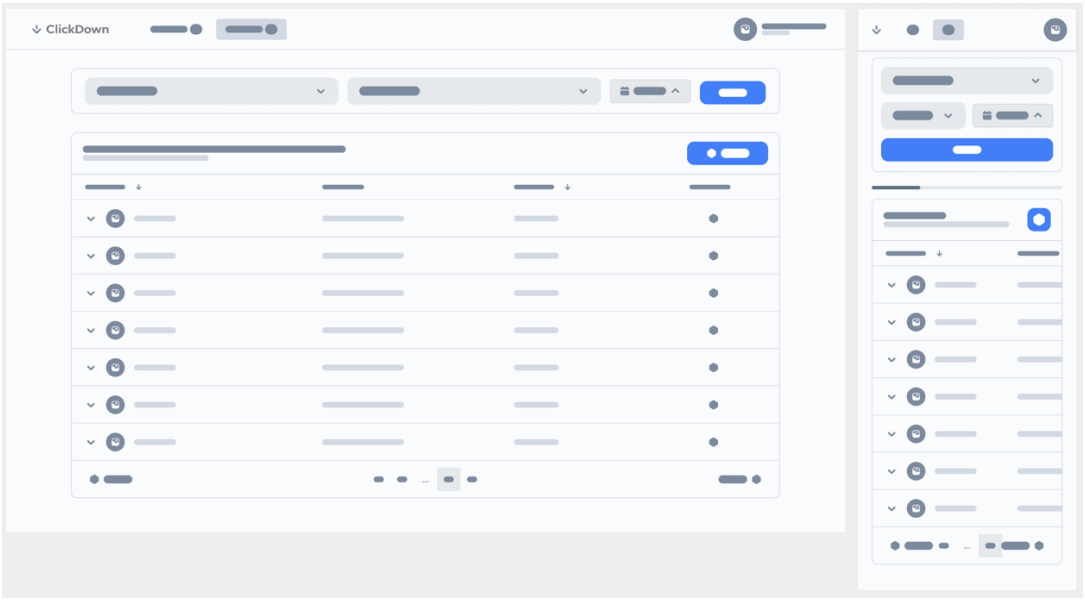

Figure 22: Accountant View Collapsed (Desktop and Mobile)

Source: own work (2025)

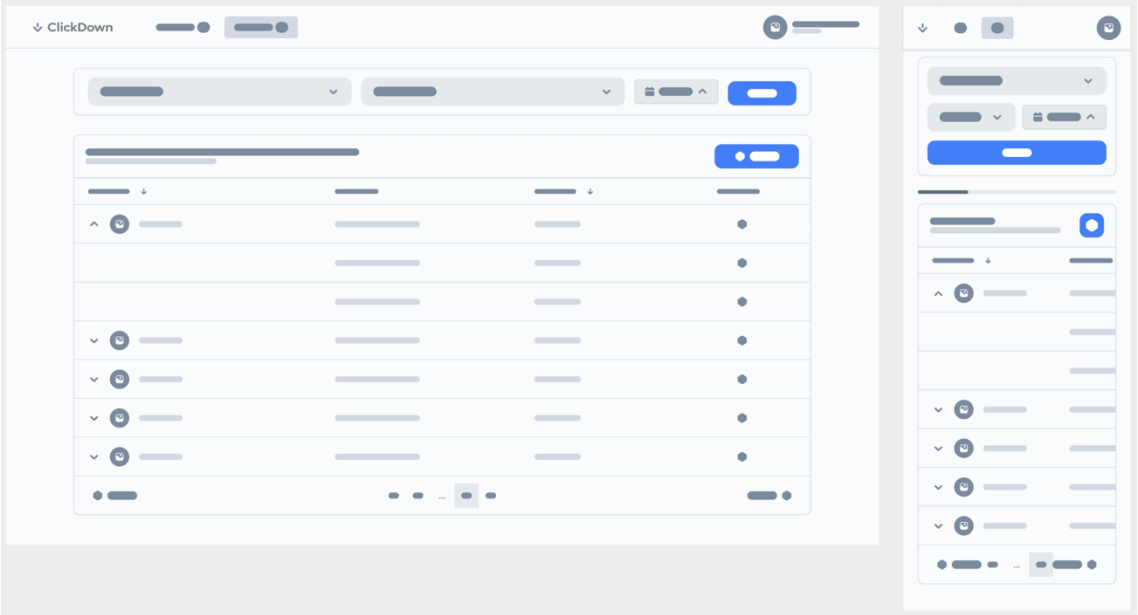

Figure 23: Accountant View Expanded (Desktop and Mobile)

Source: own work (2025)

The interface extends system functionality to aggregated organizational reporting. Structural consistency with the Worker View is maintained to reduce cognitive friction when switching roles.

The upper section includes filtering mechanisms for freelance worker selection, client selection, and month specification. An explicit Apply action is required to execute filter changes, preventing unintended automatic reload and providing clear feedback when data updates occur. Selecting "All" resets individual checkbox selections to avoid ambiguous filter states.

Below the filtering section, the table aggregates records at the freelance worker level. In collapsed state, each row displays freelance worker name, summarized client indication, total hours, and a status indicator. Expanding a row reveals client-level breakdown and associated hour values. Status indicators visually signal the presence of unassigned or potentially problematic records, enabling efficient anomaly detection.

As in the Worker View, a "Last updated" timestamp is included to communicate data freshness. Since synchronization depends on API updates and rate-limit restrictions rather than continuous real-time refresh, explicit feedback about data recency prevents incorrect assumptions about system state and supports informed financial decision-making.

The low-fidelity stage established the structural foundation of ClickDown, defined interaction logic for both user roles, and validated data grouping mechanisms required for workflow optimization at Allodium Games. By focusing exclusively on layout, hierarchy, and behaviour rather than visual styling, the phase ensured that usability principles were embedded into the system architecture while respecting technical integration constraints. The resulting structure demonstrates how API-driven limitations function not as barriers, but as drivers shaping transparent and reliable interface design.

2.4 Usability testing

After completing the low-fidelity wireframes, the next step in the design process was to evaluate them with real users before moving on to high-fidelity design. The phase aimed not at running a large formal study, but at finding out whether the structure, navigation, and layout of the proposed interface made sense to the people who would actually use it. Identifying problems at the wireframe stage is considerably more efficient than finding them after a polished prototype already exists, since changes at an early stage require far less rework (Nielsen, 1993). The findings from the phase were used to define a set of targeted design improvements that are described in Section 2.4.5 and applied in the high-fidelity prototype in Section 2.5.

2.4.1 Testing approach

The sessions were conducted as low-fidelity walkthroughs using a think-aloud protocol. In practice, each participant was shown the wireframe screens in Figma presentation mode, which is a full-screen view that hides all editing panels and displays the design as a clean, static layout. The facilitator, who was the author of the thesis, navigated between screens manually in response to what the participant said they would do. For example, if a participant pointed at a button and said they would click it, the facilitator advanced to the screen that would logically follow the action. The technique is known as a wizard of oz prototype walkthrough (Rudd et al., 1996) and is a standard approach for gathering usability feedback before a clickable prototype has been built.

Participants were asked to think out loud throughout each session, meaning they were encouraged to say what they were looking at, what they expected to happen, and what was confusing or unclear. The protocol makes it possible to observe not just whether a task is completed, but where hesitation or misunderstanding arises. The facilitator did not explain how any part of the interface worked during the tasks, as providing guidance would have influenced the observations. Notes were taken during each session, and the identified issues were reviewed and rated after all sessions were completed.

It is worth being explicit about what the participants were and were not doing. Because the wireframes were static, participants did not physically click anything. Instead, they pointed at elements on screen or described what they would interact with, while the facilitator noted whether their intention matched the intended interaction. The approach is sufficient for identifying the most common types of usability problems, including unclear labels, missing affordances, ambiguous layout, and navigation structures that do not match user expectations. As Nielsen (1993) notes, the majority of usability issues can be detected through observation alone, even without functional interaction.

The severity of each issue was rated using Nielsen's (1993) four-level scale, which ranges from cosmetic problems that have no impact on task completion to critical problems that would prevent a user from proceeding. The rating was used to prioritise which issues needed to be addressed before the high-fidelity design phase.

2.4.2 Participants

Three participants were recruited from within Allodium Games. The sample size aligns with Nielsen's (1993) recommendation that three to five participants are sufficient to uncover the majority of usability problems in a small-scale evaluation. More importantly, all three participants were actual members of the tool's intended user base: they use ClickUp for time tracking in their daily work and would be expected to use ClickDown once it is deployed. Testing with users who have genuine familiarity with the work context produces more relevant observations than testing with people who are unfamiliar with the domain.

Two participants represented the freelance worker role, as the freelance worker role represents the most frequent interaction type in the system. One participant represented the accountant role, reflecting the role defined in Section 2.2.1. Before each session began, participants were told that the purpose was to evaluate the design and that any difficulty they encountered was useful information, not a reflection of their own abilities.

Table 10: Overview of usability testing participants

Participant	***Role***	***Daily tools***	***Testing context***
P1	Developer (freelance worker role)	ClickUp (daily), Git	Worker View tasks
P2	Designer (freelance worker role)	ClickUp (daily), Adobe Illustrator	Worker View tasks
P3	Financial analyst (accountant role)	CSV exports, Excel, ClickUp (ad hoc)	Accountant View tasks

Source: own work (2025)

The combination of roles and tool familiarity ensured that observations from the sessions would reflect actual usage patterns rather than generalized responses from people unfamiliar with the work context.

2.4.3 Task Scenarios

Rather than asking participants to explore the prototype freely, each session was structured around a short scenario that gave the interaction a realistic context. Goal-oriented tasks produce more natural behaviour than open exploration and better reflect how the tool would actually be used (Nielsen, 1993). The scenarios were designed to reflect the core workflows identified in Section 2.1 and the functional requirements defined in Section 2.2.2.

For P1 and P2, who represented the freelance worker role, the following scenario was read aloud at the start of the session: "It is the end of November and you want to check how many hours you worked this month and whether your time has been logged against the right clients, before the accountant closes the month."

For P3, who represented the accountant role, the scenario was: "It is the beginning of December, and you need to review the November hours across the whole team so you can prepare invoices. You want to check the totals per freelance worker and per client, find any entries that are missing a client, and export the data."

Within each scenario, participants were asked to work through a set of specific tasks, which are listed in Table 11 below.

Table 11: Task scenarios used during usability testing

Task ID	***Description***	***Role***	***Success criterion***
T1	Open the application and authenticate	Both	Reaches the main view without confusion
T2	Find the total number of hours worked in November	Freelance worker	Correctly reads the value from the summary card
T3	Identify which client has the most logged time	Freelance worker	Points to the correct row in the client table
T4	Change the report to show October instead of November	Freelance worker	Identifies the month selector and describes how they would use it
T5	Find the Unassigned category and describe what it contains	Freelance worker	Notices the row and attempts to interact with it
T6	Locate and trigger the export function	Both	Points to the Export button in the correct location
T7	Switch to the Accountant View	Accountant	Identifies and points to the correct tab in the navigation
T8	Filter the report to show only one freelance worker	Accountant	Describes using the dropdown and explains the expected result
T9	Find entries where the client could not be identified	Accountant	Notices the unassigned rows and the associated visual indicator
T10	Expand a freelance worker row to see the per-client breakdown	Accountant	Points to the correct interactive element in the row

Source: own work (2025)

The tasks were designed to cover the primary interactions in each role without guiding participants toward any particular behaviour or sequence, so that hesitation and uncertainty could surface naturally during the walkthrough.

2.4.4 Task scenarios

The sessions produced a number of clear observations that helped identify both what was working well in the wireframes and where specific improvements were needed. The paragraphs below discuss the main findings from all three sessions, organised by screen and by the severity of the issues observed.

The login screen (T1) presented no difficulties for any of the three participants. All of them identified the Authenticate via ClickUp button immediately and understood that authentication was handled through ClickUp. The minimal layout of the screen, showing only the logo, a short tagline, and the action button, was sufficient to communicate its purpose without additional explanation.

In the Worker View, both freelance worker participants quickly identified their total worked hours (T2) and the top client (T3). The summary card, which displayed a clear numeric value in

a visually distinct component, was described by P1 as immediately clear. The client table was also well understood: both participants correctly read the layout as a ranked list of clients with proportional time bars and exact hour values. The finding is consistent with Few's (2006) argument that combining visual encoding with precise numeric data improves how quickly users can extract meaning from a reporting interface.

When asked how they would change the reporting period to October (T4), both participants identified the month selector button in the top right of the header. The wireframe screen showing the open state of the selector was then presented. P1 looked at the dropdown layout and asked: "Would I pick a specific day, or just the month?" The visual design of the picker did not clearly communicate that only month-level selection was available. The presence of a date-like grid structure in the dropdown created an expectation that day-level selection might be possible, which is a false affordance in the current design (Norman, 2013).

The most significant finding from the freelance worker sessions concerned the Unassigned row in the client table (T5). When shown the Worker View screen, P1 looked over the table but did not initially notice the chevron next to the Unassigned row. After a moment, P1 pointed at it and asked: "Oh, is this one expandable?" — showing that the affordance was not immediately obvious, even once the row was noticed. P2 also saw the Unassigned row but did not attempt to interact with it at all, commenting: "I could see the row was there, but I did not think it was something I could click." The problem lies in the signalling of interactivity. According to Norman (2013), an interface element should communicate what action it invites rather than requiring users to discover it through trial and error. Since the Unassigned row is an important part of the self-verification workflow, as it allows freelance workers to check what time could not be matched to a client, the finding was treated as a high-priority issue.

The export function (T6) was located without difficulty by both freelance worker participants, who pointed to the Export button in the section header. P2 asked whether the export would be a CSV file or a PDF, which indicates that the button label alone does not communicate the format of the output. The issue is minor since it does not prevent task completion, but it does create uncertainty in a context where the file format matters for downstream processing.

In the Accountant View, P3 was first shown the Worker View and asked to find the section that would show the full team report (T7). P3 looked at the navigation bar and said: "I see Worker, but where is the team report? Is it somewhere else?" After a few seconds, P3 identified the second tab and pointed to it. The tab label as designed did not make it immediately clear to a user in the accountant role that the tab existed for them. The navigation issue is relatively minor but one that could slow down first-time use.

Applying the freelance worker filter (T8) was straightforward once P3 was inside the Accountant View. P3 correctly identified the dropdown, described selecting a single freelance worker, and explained that they would expect the table to update to show only that person's data. The intended behaviour of the filter matches what P3 described, suggesting that the filter bar layout was clear enough for the accountant role.

Identifying rows with missing client assignments (T9) was more problematic. When shown the Accountant View, P3 noticed that certain rows had a small visual marker next to them and said: "I can see something is different about these rows, but I am not sure what it means." The status indicator in the wireframe was a small dot, present and visible but not labelled or explained

anywhere on the screen. Since identifying unassigned time entries is one of the central tasks of the accountant's monthly workflow as described in Section 2.1.3, the finding was treated as a high-priority issue. The indicator needs to communicate its meaning without relying on the user already knowing what it represents.

When asked where they would click to expand a freelance worker row (T10), P3 pointed to the avatar icon rather than the expand chevron, commenting: "I would click here on the person icon." The chevron was positioned close to the avatar but was visually smaller and less prominent, which caused P3 to target the wrong element. The root cause is a click target issue: even in a static walkthrough where no actual click is required, participants naturally point to the most visually salient element, which reveals which element they perceive as interactive.

The main issues identified across the three sessions are summarised in Table 12, together with their severity ratings and the participants who encountered them.

Table 12: Summary of usability issues identified during testing, rated using Nielsen's (1993) severity scale

Issue	***Description***	***Participants affected***	***Severity (Nielsen, 1993)***
I1	The Unassigned row is not visually signalled as interactive; the expand chevron is too small and not distinct enough to invite interaction	P1, P2	3 – Major
I2	The status indicator in the Accountant View is visible but has no label or explanation; users can see something is different but cannot determine what it means	P3	3 – Major
I3	The Worker tab label does not communicate to accountant-role users that a separate team report view exists	P3	2 – Minor
I4	The open state of the month picker contains a date-like grid structure that suggests day-level selection may be possible, creating a false affordance	P1	2 – Minor
I5	The expand chevron in the Accountant View rows is less visually prominent than the avatar icon placed next to it; users point to the wrong element	P3	2 – Minor
I6	The Export button does not indicate what format the output file will be in, creating uncertainty for users who need a specific format	P2, P3	1 – Cosmetic

Source: own work (2025)

The distribution of severity ratings — two major issues, three minor, and one cosmetic — indicates that the wireframes were structurally sound but required targeted refinement in the areas of affordance signalling and information labelling. No issues were rated as critical, meaning no participant was unable to complete a task entirely.

2.4.5 Design improvements

The findings from Section 2.4.4 were used to define a set of design improvements that were applied during the transition to the high-fidelity prototype. Each improvement is directly connected to a specific issue observed during testing and is grounded in the design principles discussed in Section 1.2.2. The changes are described below in order of priority.

The most important change addresses the discoverability of the Unassigned row interaction (I1). In the low-fidelity wireframes, the row looked almost identical to the regular client rows, with only a small chevron indicating it could be expanded. To make the interaction more obvious in the high-fidelity design, the row is given a distinct visual treatment. A labelled expand button reading "Show tasks" is introduced inside the row, and a hover state is added so that users can see the element is interactive before they attempt to interact with it. The change applies Norman's (2013) principle of signifiers, which holds that interactive elements should communicate their function visually rather than requiring users to discover it through exploration. Because reviewing unassigned time is part of the core self-verification workflow for freelance workers, making the interaction clear is essential for the usefulness of the screen.

The second major change resolves the unexplained status indicator in the Accountant View (I2). The small dot used in the wireframes to flag rows with missing client data is replaced in the high-fidelity design with an explicit inline badge containing a warning symbol and the label "Unassigned." A tooltip added on hover explains what the badge means: "This time entry could not be matched to a known client." Without the change, the accountant would have no reliable way to identify data quality issues in the monthly report, which is one of the central problems described in Section 2.1.3.

To make the navigation clearer for users in the accountant role (I3), the tab labels are updated in the high-fidelity prototype. The label "Worker" is changed to "My Report," and the second tab is explicitly labelled "Team Report." The change removes the ambiguity observed during testing and makes the role-based structure of the application immediately understandable to a new user, without introducing any additional navigation complexity.

The month picker is updated to remove the date-grid appearance that caused P1 to question whether day-level selection was available (I4). In the high-fidelity design, the picker shows only month and year labels, with no grid structure that could imply a finer level of selection. The approach is consistent with Norman's (2013) principle of constraints, which states that the design should make only the possible actions visible rather than presenting options that do not apply.

To resolve the click target issue in the Accountant View (I5), the entire freelance worker row is made interactive in the high-fidelity design, rather than only the chevron icon. Clicking anywhere in the row expands or collapses the client breakdown. The model is more forgiving and eliminates the problem of users targeting the avatar instead of the chevron; it is also more consistent with how expandable rows are handled in tools that participants already use, such as Notion and Linear.

The Export button is relabelled "Export as CSV" in the high-fidelity design (I6). The relabelling removes the uncertainty observed with P2 and P3 regarding the file format. The issue is cosmetic and did not prevent task completion, but the change is small and straightforward and directly

addresses an observed source of confusion — particularly relevant for the accountant, who needs to know the format before exporting.

The six improvements represent targeted, evidence-based changes to the interface. None of them alter the information architecture or the core logic defined in Section 2.3. They refine how existing elements are presented and labelled, based on direct observation of how real users interpreted the wireframes. Usability testing at the low-fidelity stage serves precisely to identify the specific points where the current design falls short of communicating clearly, rather than to rethink the design as a whole. The implementation of the changes in the high-fidelity prototype is described in Section 2.5.

2.5 Final prototype and design system integration

Following the usability testing described in Section 2.4, the design process moved to the high-fidelity phase. The phase involved two parallel tasks: developing the ClickDown design system and applying it to produce fully detailed interface screens. The structural decisions and interaction logic were already established in the low-fidelity prototype; the purpose of the phase was to define every visual detail in a consistent and documented way, and to incorporate the six design improvements identified in Section 2.4.5.

Section 2.5 describes the design system first, as it forms the visual foundation of the prototype. It then presents the high-fidelity screens, focusing on what changed compared to the low-fidelity wireframes and how the design system was applied. The section ends with a summary of how the functional requirements from Section 2.2.2 are covered in the completed prototype.

2.5.1 Design system

A design system is a structured collection of reusable design decisions, including colours, typography, spacing values, and component definitions, that ensures visual and behavioural consistency across all parts of a product. According to Vesselov and Davis (2019), design systems reduce ambiguity by providing a shared visual language that can be applied consistently across different screens and states. For ClickDown, creating a dedicated design system was necessary given the complexity of the interface: with two user roles, multiple views, and a range of interactive components, maintaining consistency without a defined system would have been difficult.

The ClickDown design system covers four areas: colour, typography, spacing and shape, and components. Each area is described below. The complete design system is available here.

Colour: The colour palette is based on a limited set of blue-grey tones, known in design practice as the Slate scale, combined with a single brand accent colour in blue (#2563EB). The approach follows Few's (2006) recommendation to restrict the colour palette so that each colour carries a specific meaning rather than serving a decorative function.

Background and surface colours create a three-level depth hierarchy. The navigation bar uses the darkest tone (#0F172A), page backgrounds use a near-white shade (#F8FAFC), and content cards use pure white (#FFFFFF). The structure directs attention toward the data area. Text colours range from a near-black primary tone (#0F172A) for headings and key values to a muted

secondary tone (#94A3B8) for less important information such as timestamps and column headers, following the visual hierarchy principles described by Few (2006, pp. 74–79).

Two semantic colour palettes are used to communicate the status of time entries. A red-toned palette is reserved for entries flagged as unassigned or overdue, while a green-toned palette is used for correctly categorised records. Using colour to encode status supports rapid interpretation without relying on text labels alone, which is consistent with the principle of preattentive processing described by Few (2006). An additional set of eight avatar colour combinations is used to distinguish individual team members across the Team Report view.

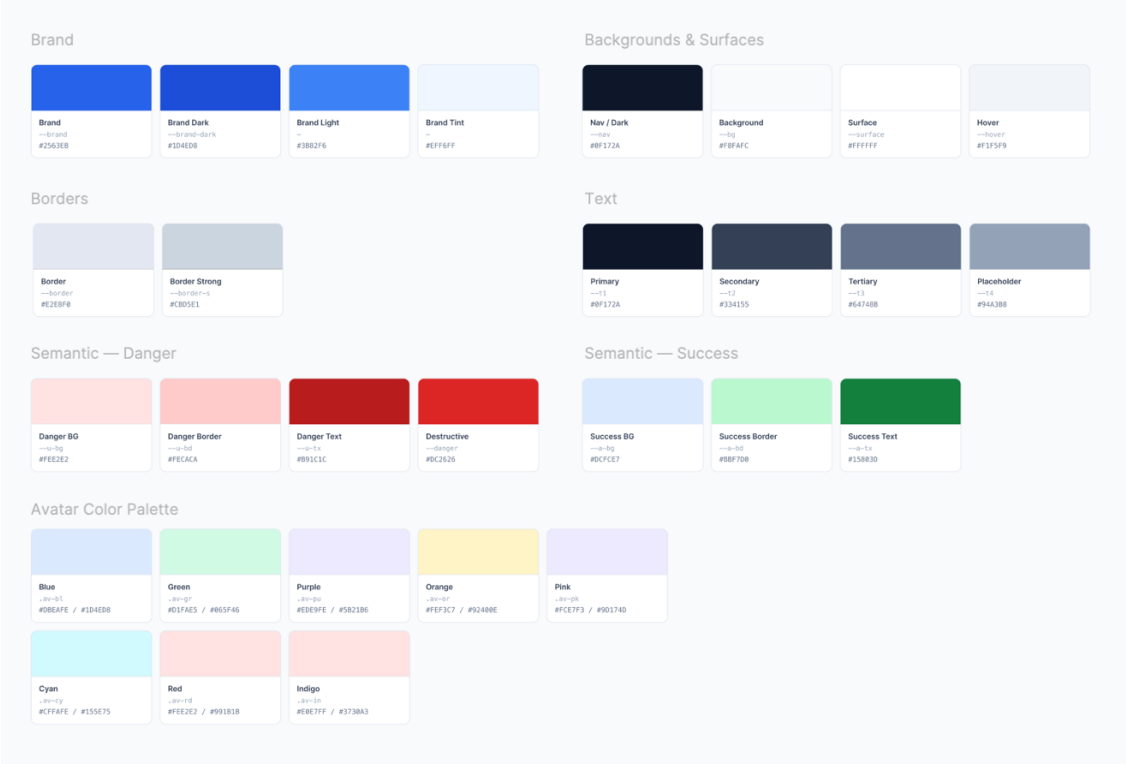

Figure 24: ClickDown design system, colour tokens and semantic palettes

Source: own work (2026)

Typography: ClickDown uses a single typeface, Inter, across all screens and components. Inter is a sans-serif typeface designed for screen readability, which makes it suitable for a data-dense interface. Using one typeface throughout the application avoids the cognitive disruption that can result from mixing typefaces (Norman, 2013).

The typographic scale ranges from 28 pt for the main stat value in the hours card down to 10.5 pt for table column headers. Weight is used systematically to communicate importance: bold (700) is applied to headings, key data values, and page titles; SemiBold (600) is used for card titles, uppercase column headers, and badge labels; Medium (500) is the default weight for interactive elements and table body content. The structure ensures that visual weight reflects informational priority, reducing the effort required to locate relevant data (Few, 2006).

Spacing and shape: Spacing follows a 4-pixel base grid, with most padding and gap values being multiples of four or six. The layout is therefore visually consistent and ensures that related elements appear grouped without the need for explicit dividers. Border radius values vary by element type: smaller values (4–7px) are used for compact interactive elements such as buttons

and inputs, while larger values (10–12px) are used for cards and modal dialogs. The variation communicates the visual weight and role of each element, consistent with Norman's (2013) principle that the appearance of an element should reflect how it functions.

Shadows are defined across four levels of intensity, ranging from a minimal shadow for standard cards to a stronger shadow for modal dialogs. The depth cues reinforce the spatial structure of the interface and make overlay elements such as the month picker and confirmation dialogs clearly distinguishable from the background content.

Components: The component library defines the visual rules for every reusable element in the interface. The navigation bar uses the darkest colour token (#0F172A) as its background and contains the logo, tab navigation, and user identity area. The active tab is marked with a visible underline indicator. Buttons are defined in four variants: a primary dark-fill button for main actions, a brand blue-fill button for key calls to action, a secondary outlined button for export and navigation actions, and a ghost button for low-priority actions. Each variant also has a small-size option for compact contexts. Consistent button styling ensures that interactive elements are recognisable and predictable (Norman, 2013).

Badges are pill-shaped status indicators used in the Team Report. 'Assigned' badges use the green semantic palette and 'Unassigned' badges use the red semantic palette. Avatars are circular, initials-based identifiers available in three sizes (40px, 32px, and 24px) and eight colour combinations, supporting consistent visual identification of team members. Cards are available in two types: the Hours Card for displaying a single aggregated value with a copy button, and the Table Card as a container for data tables with a header, export button, and pagination footer. The filter bar, present in the Team Report view, groups three multi-select dropdowns with an Apply button into a single horizontal row. The month picker is a compact dropdown showing a year selector and a grid of abbreviated month names. Modals are centred confirmation dialogs displayed above a semi-transparent overlay.

With regard to interactive behaviour, the design specifies short response times for all state changes. It is recommended that hover and active states across buttons, rows, and tabs are implemented with transitions of approximately 150 milliseconds. For the progress bar, a slightly longer fill animation of around 600 milliseconds is suggested to provide a clear visual cue when report data loads. The values are design recommendations for the development phase and may need to be adjusted based on the technical constraints of the implementation stack. As both Few (2006) and Nielsen (1993) note, motion in data interfaces should serve comprehension rather than act as decoration.

The component set is illustrated in Figure 25, showing the navigation bar with tab structure and user identity area, the full range of button variants including primary, brand, destructive, secondary, and ghost styles, the filter bar in both collapsed and open states, and the month picker panel.

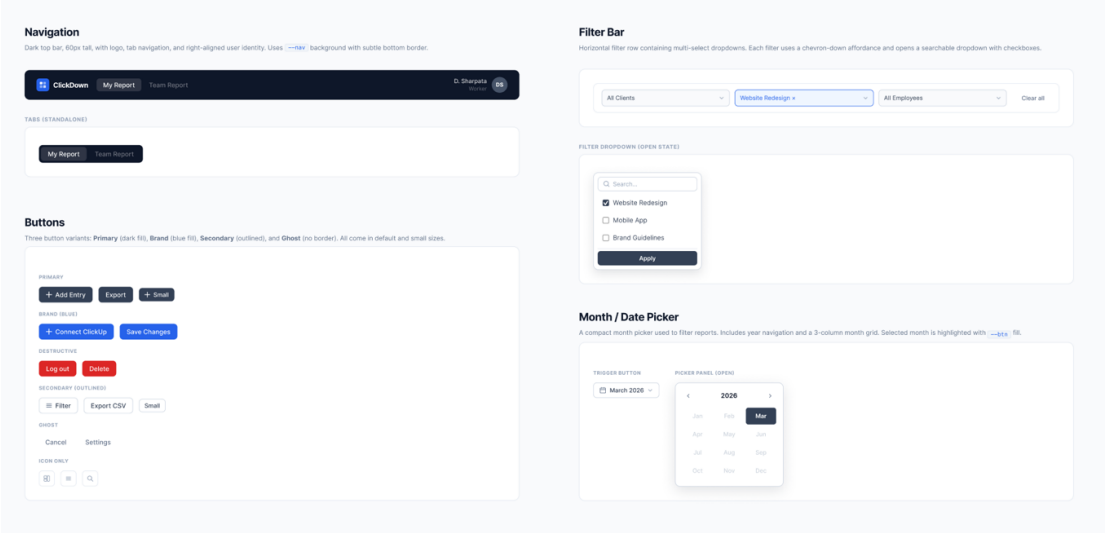

Figure 25: ClickDown design system, selected components

Source: own work (2026)

The components shown cover the complete set of interactive elements used across both views. Every variant and state is defined at the system level, ensuring that the same element behaves and appears identically regardless of which screen it appears on.

2.5.2 High-fidelity prototype

The high-fidelity prototype was produced by applying the design system defined in Section 2.5.1 to the wireframe structure from Section 2.3.4. The screen layout, information architecture, and interaction logic described in Section 2.3 remained unchanged. Section 2.5.2 focuses on what was added or modified at the high-fidelity stage: the visual design choices and the six improvements identified during usability testing in Section 2.4.5. The prototype is available here and also in the Appendix C.

The login screen received a full visual treatment using the dark navigation colour as the background and the brand blue button as the primary call to action. The structure remains identical to the wireframe, as no usability issues were identified for the screen during testing.

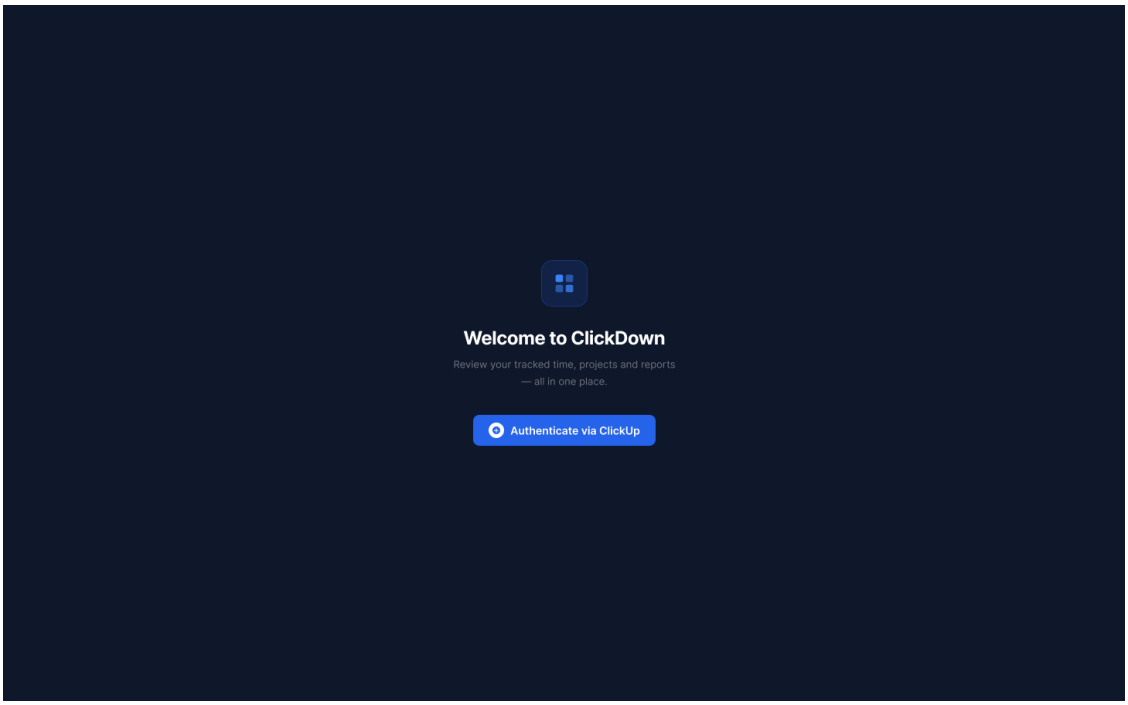

Figure 26: High-fidelity login screen, desktop
Source: own work (2026)

In the My Report view, the design system tokens were applied to all elements: background, surface, and text colour tokens establish the visual hierarchy, and the Inter typeface is used consistently across the page title, summary card, and table. The progress bars in the client table use the designated fill colour, and the brand blue is applied to the month selector button and the export action. Four of the six usability improvements are reflected in the view. The Unassigned row (issue I1) is now visually distinct from regular client rows, with a lighter background fill and a clearly labelled 'Show tasks' button that communicates interactivity before the user attempts to activate it, addressing the signifier problem described by Norman (2013). The month picker (issue I4) was redesigned to show only a grid of month names with no day-level cells, removing the false affordance present in the wireframe. The export button (issue I6) is labelled 'Export as CSV' instead of the generic 'Export', making the output format clear before the action is triggered. The tab label (issue I3) was changed from 'Worker' to 'My Report', which is described further below.

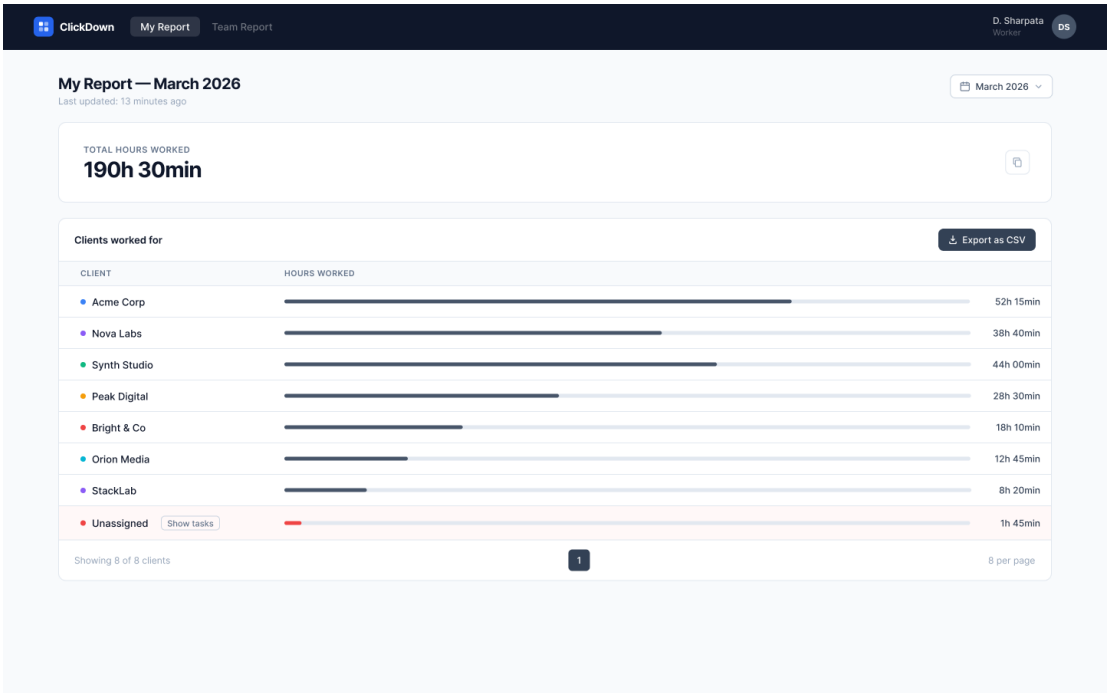

Figure 27: My Report view, default state, desktop resolution

Source: own work (2026)

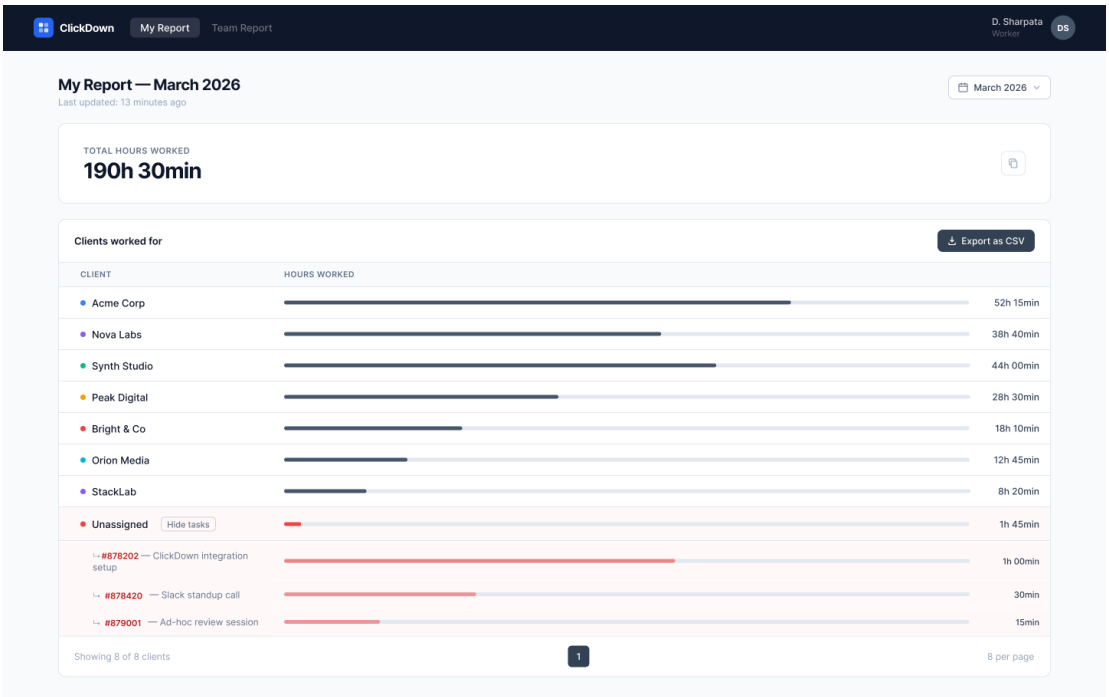

Figure 28: My Report view, Unassigned row expanded, task-level breakdown visible

Source: own work (2026)

The month picker is shown as a separate state to illustrate the redesigned panel. The grid contains only abbreviated month names with no date-level structure that could suggest a finer selection granularity.

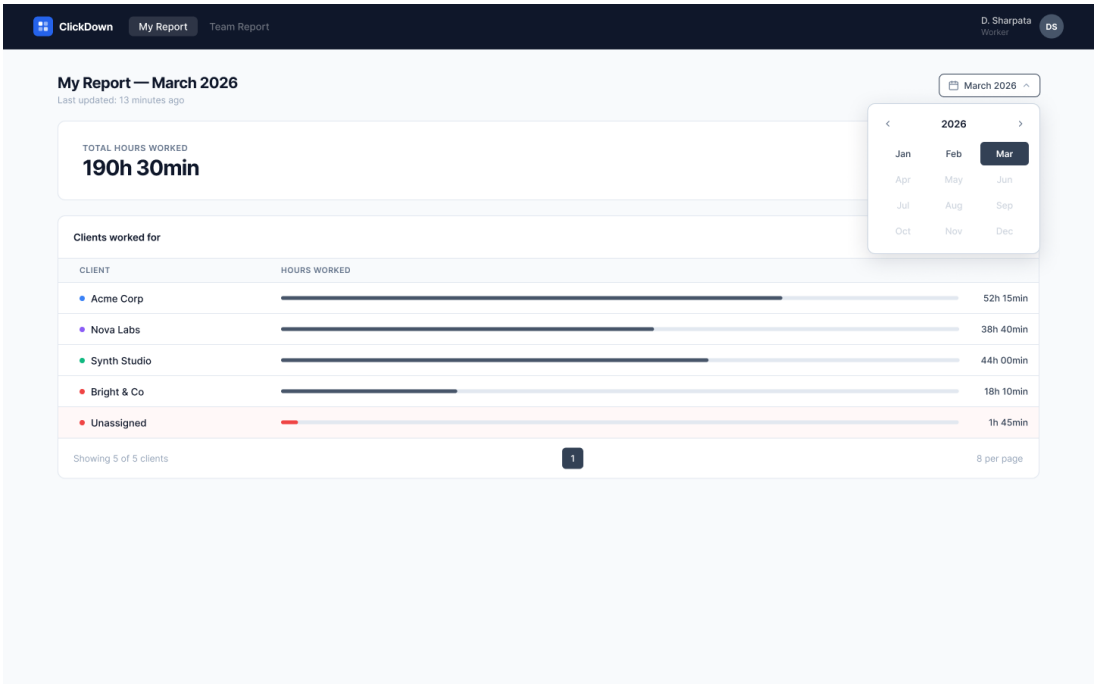

Figure 29: Month picker open state

Source: own work (2026)

The Team Report view received the most significant visual changes at the high-fidelity stage. The status indicator for unassigned entries (issue I2) was changed from a small unlabelled dot to a fully styled 'Unassigned' badge using the red semantic colour palette. For implementation, it is recommended that hovering over the badge displays a tooltip explaining that the freelance worker has time entries that could not be matched to a known client, so that users do not need to interpret the indicator from context alone. The tab label (issue I3) was updated to 'Team Report', creating a clear contrast with the renamed 'My Report' tab and making the role-based structure of the application immediately understandable.

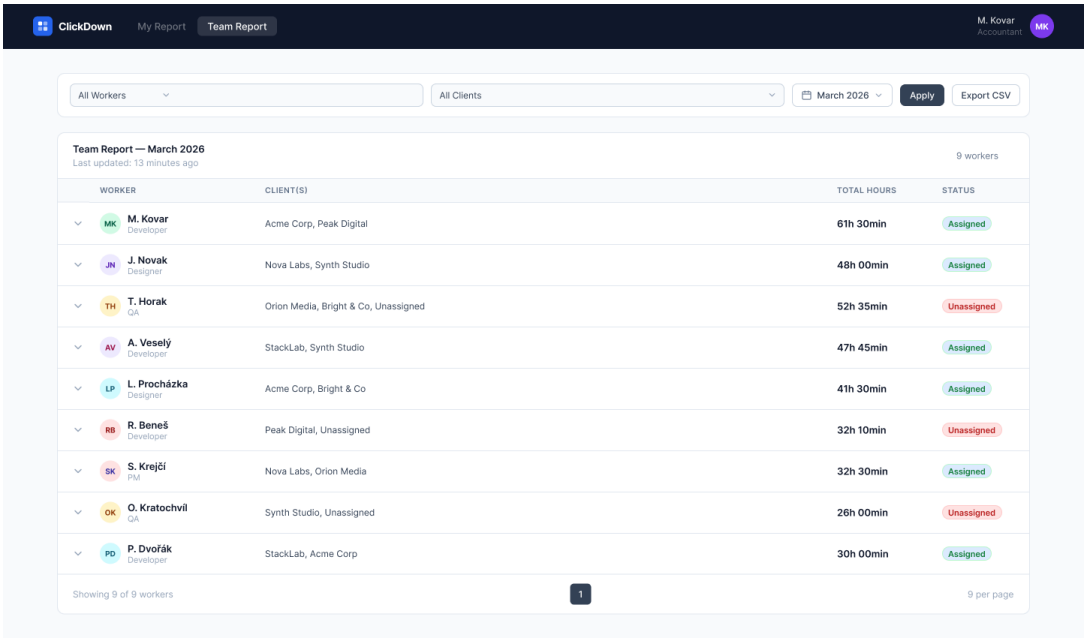

Figure 30: Team Report view, default state, desktop resolution

Source: own work (2026)

Figure 31 shows the filter bar in use, with the freelance worker dropdown open. The searchable checkbox list allows the accountant to select one or multiple freelance workers before applying the filter, limiting the displayed data to the selected subset.

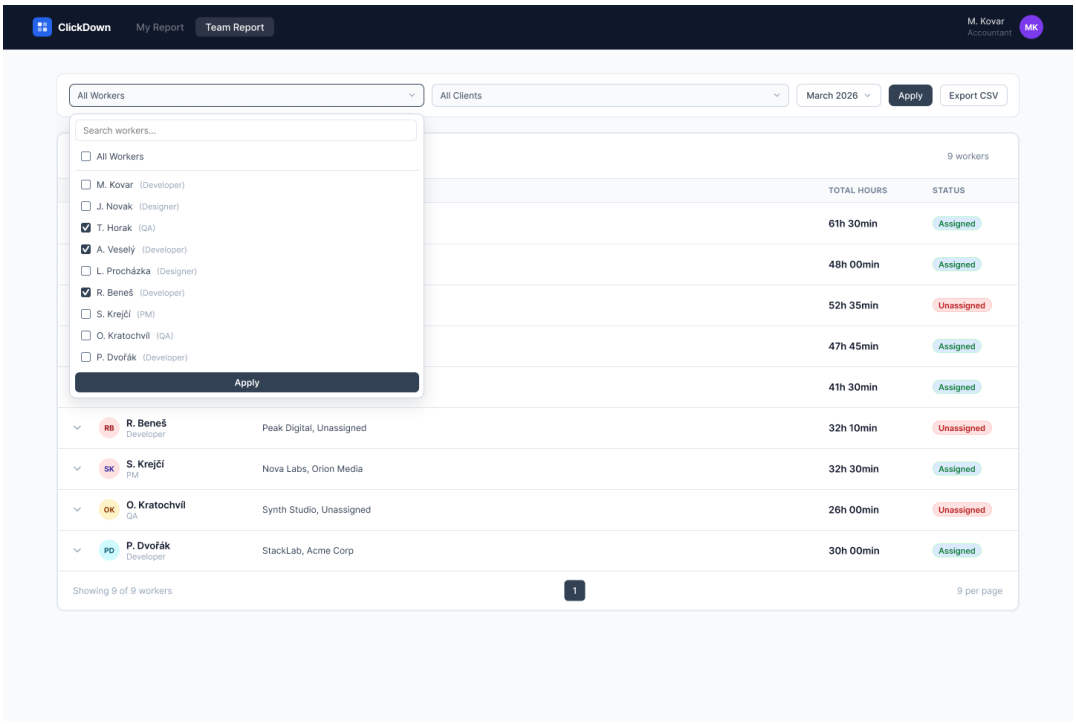

Figure 31: Team Report view, freelance worker filter open state

Source: own work (2026)

The expand interaction for freelance worker rows (issue I5) was redesigned so that the entire row is the clickable area, replacing the small chevron that participants consistently missed during testing. Avatars in the table use the defined colour palette to distinguish team members visually.

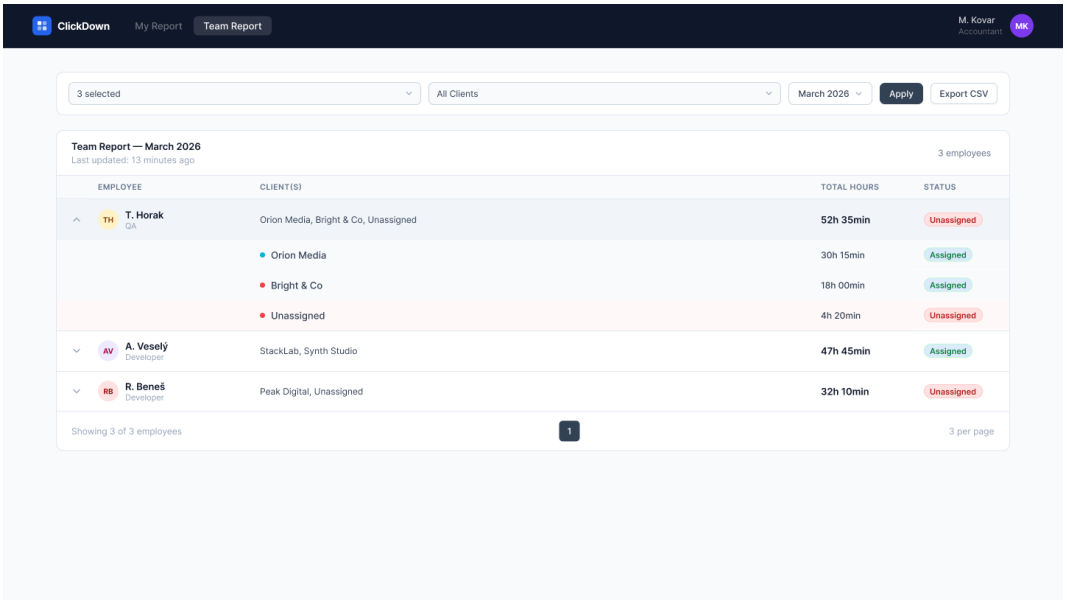

Figure 32: Team Report view, freelance worker row expanded, per-client breakdown visible

Source: own work (2026)

The logout confirmation modal is shown as a separate state. It uses the destructive red button variant defined in the design system and appears above a semi-transparent overlay, making the temporary interruption of navigation visually clear.

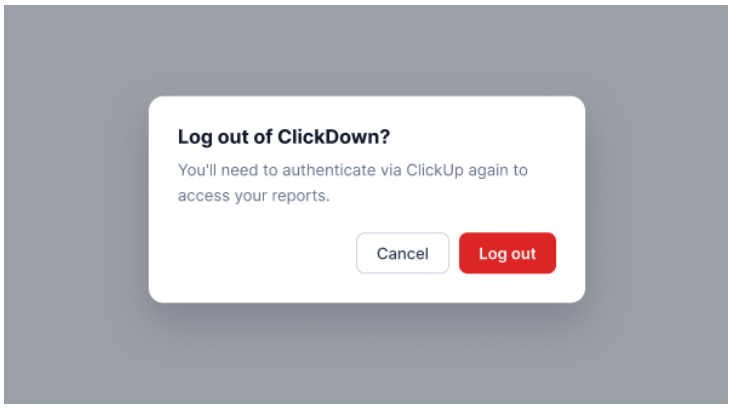

Figure 33: Logout confirmation modal

Source: own work (2026)

Together, the six improvements from Section 2.4.5 address every major usability issue identified during testing without changing the underlying structure of the interface. The changes are targeted, evidence-based, and grounded in the design principles discussed in Section 1.2.2.

The prototype is presented in a format close to a working application, but as is standard for Figma-based prototypes, some interactions are represented through defined states rather than live behaviour. The date picker shows default, selected, and disabled states, but selecting a different month does not update the displayed data. Filters open and demonstrate the selection interaction, but the table does not refresh after applying them. In the Team Report, the expandable row interaction is demonstrated on the T. Horak row specifically, due to how screen links are structured in Figma, while the intended behaviour applies to all rows as described in Section 2.4.5.

Simulating live data at the prototype stage would add complexity without contributing to the evaluation of layout, visual design, or interaction clarity. The static dataset used throughout is sufficient to validate all design decisions. The prototype represents a complete specification for a development team. A further round of usability testing before production implementation is recommended to confirm that the design improvements from Section 2.4.5 work as intended with a wider group of users.

2.5.3 Requirements coverage

The completed high-fidelity prototype covers all ten functional requirements defined in Section 2.2.2. Table X provides an overview of how each requirement is addressed in the final design.

Table 13: Coverage of functional requirements in the high-fidelity prototype (requirements defined in Table 8, Section 2.2.2)

ID	*Covered by in prototype*
FR1	My Report – Hours Card
FR2	My Report – client table with progress bars
FR3	Role-based tab access; Member-level users can only open My Report
FR4	Unassigned row (My Report); Unassigned badge (Team Report)
FR5	Team Report – aggregated table
FR6	Team Report – filter bar with Apply button
FR7	Team Report – per freelance worker and per-client hour totals
FR8	Export as CSV button (My Report and Team Report)
FR9	Team Report – default state (no task rows visible)
FR10	Login screen (ClickUp SSO); no data entry or edit elements throughout

Source: own elaboration (2026)

FR1 and FR2 require the system to display the total hours worked and a breakdown by client for the authenticated user. Both are addressed by the My Report view. FR3, which restricts freelance workers to viewing only their own data, is enforced through role-based tab access tied to ClickUp workspace membership. FR4, requiring the system to flag entries with missing client assignments, is implemented through the Unassigned row in the My Report view and the red badge in the Team Report.

FR5, FR6, and FR7 are addressed by the Team Report, which shows an aggregated monthly overview across all freelance workers and clients, supports filtering by month, freelance worker, and client, and provides totals that can be used directly for invoicing preparation. FR8 is covered by the 'Export as CSV' function available in both views. FR9, which requires a high-level overview for management users, is supported by the default state of the Team Report. FR10, covering authentication via ClickUp SSO and read-only operation, is reflected in the interface design: the application contains no data entry or editing elements.

Non-functional requirements are addressed throughout the design. Data freshness is communicated through the 'Last updated' timestamp shown in both views. Reliability is supported by the visible Unassigned indicators that surface data quality issues explicitly. Security is maintained through role-based access derived from ClickUp membership levels. Usability is addressed by applying the design principles of Few (2006), Norman (2013), and Nielsen (1993) at each stage of the process. Maintainability is supported by the token-based structure of the design system, which allows changes to individual values to propagate consistently across all components.

Together, the design system and the high-fidelity prototype form the main practical output of the thesis. The completed design addresses all identified requirements and provides a detailed specification that can be used as the basis for implementation of the ClickDown reporting tool.

Conclusion

The goal of this thesis was to improve project management workflows at Allodium Games by designing a dedicated web application. The work followed the human-centred design process defined by ISO 9241-210 (2019), which meant starting from real user needs rather than assumptions.

Interviews with the chief technology officer and the company accountant showed that the studio's existing ClickUp setup handled core project coordination well enough. The real problem was at the reporting level: freelance workers had no reliable way to check their monthly hours, and the accountant had to prepare invoices manually using CSV exports. Findings from stakeholder interviews informed all subsequent design decisions.

From there, functional and non-functional requirements were defined, and the design moved through information architecture, user flows, and low-fidelity wireframes. The wireframes were evaluated using a Wizard of Oz usability test with three participants, which surfaced specific issues around navigation, filtering, and the handling of unassigned time entries. Usability findings were used to refine the design before moving to the high-fidelity prototype, which was built on top of a complete design system covering colour, typography, spacing, and reusable components.

The final prototype covers the two core views of the application: My Report for individual users to track their own hours, and Team Report for managers and the accountant to oversee the full team's time entries, both supporting expandable row interactions for additional detail. Since ClickDown works as a reporting layer on top of the existing ClickUp API, the studio can start using it without changing anything in their current setup.

The main outcome is a fully specified design, grounded in usability research, with a design system providing a solid foundation for consistent implementation and future updates.

Looking ahead, ClickDown could be extended with features like automated monthly report scheduling, proactive alerts for unassigned entries, and more granular role-based views. A further round of usability testing before production deployment would help confirm that changes made after the first test round hold up with a wider group of users.

Bibliography

ASANA. 2025a. How Asana Works [online]. [cit. 2025-02-01]. Available at: https://help.asana.com/s/article/how-asana-works

ASANA. 2025b. Universal Reporting [online]. [cit. 2025-02-01]. Available at: https://help.asana.com/s/article/universal-reporting

ATLASSIAN. 2025a. Get Started with Jira: Comprehensive Beginner's Guide [online]. [cit. 2025-02-01]. Available at: https://www.atlassian.com/software/jira/guides/getting-started/introduction

ATLASSIAN. 2025b. Learn About Jira Software Reports & Dashboards [online]. [cit. 2025-02-01]. Available at: https://www.atlassian.com/software/jira/guides/reports-dashboards/overview

ATLASSIAN COMMUNITY. 2025. How to View All Logged Time by Users in Jira: Step-by-Step Guide [online]. [cit. 2025-02-01]. Available at: https://community.atlassian.com/forums/App-Central-articles/How-to-View-All-Logged-Time-by-Users-in-Jira-Step-by-Step-Guide/ba-p/1633245

BROWN, Tim. 2009. Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation. New York: Harper Business.

CLICKUP. 2023. ClickUp Dashboards: Customizing Your Workspace Overview [video online]. In: YouTube [online]. [cit. 2025-02-01]. Available at: https://www.youtube.com/watch?v=lqqSsEQ-tTk

CLICKUP. 2024a. Authentication [online]. ClickUp [cit. 2025-02-01]. Available at: https://developer.clickup.com/docs/authentication

CLICKUP. 2024b. ClickUp Dashboards: Visualize Your Productivity [online]. ClickUp [cit. 2025-10-06]. Available at: https://clickup.com/features/dashboards

CLICKUP. 2024c. ClickUp's MCP Server Overview [online]. ClickUp [cit. 2025-02-01]. Available at: https://developer.clickup.com/docs/connect-an-ai-assistant-to-clickups-mcp-server

CLICKUP. 2024d. Custom Fields [online]. ClickUp [cit. 2025-02-01]. Available at: https://developer.clickup.com/docs/customfields

CLICKUP. 2024e. Date Formatting [online]. ClickUp [cit. 2025-02-01]. Available at: https://developer.clickup.com/docs/general-time

CLICKUP. 2024f. MCP Server Setup Instructions [online]. ClickUp [cit. 2025-02-01]. Available at: https://developer.clickup.com/docs/connect-an-ai-assistant-to-clickups-mcp-server-1

CLICKUP. 2024g. OpenAPI Specification [online]. ClickUp [cit. 2025-02-01]. Available at: https://developer.clickup.com/docs/open-api-spec

CLICKUP. 2024h. Rate Limits [online]. ClickUp [cit. 2025-02-01]. Available at: https://developer.clickup.com/docs/rate-limits

CLICKUP. 2024i. Tasks [online]. ClickUp [cit. 2025-02-01]. Available at: https://developer.clickup.com/docs/tasks

CLICKUP. 2024j. ClickUp Timesheets: View, Track, and Review Time-Tracked Tasks [online]. ClickUp [cit. 2025-10-07]. Available at: https://clickup.com/blog/timesheet-management/

CLICKUP. 2024k. Webhook Payloads and Automation Events [online]. ClickUp [cit. 2025-02-01]. Available at: https://developer.clickup.com/docs/automationwebhookpayload

CLICKUP. 2024l. Webhooks [online]. ClickUp [cit. 2025-02-01]. Available at: https://developer.clickup.com/docs/webhooks

CLICKUP. 2024m. What is ClickUp Used for and How Does It Work? [online]. ClickUp [cit. 2025-10-06]. Available at: https://clickup.com/blog/what-is-clickup-used-for/

FEW, Stephen. 2006. Information Dashboard Design: The Effective Visual Communication of Data. Sebastopol: O'Reilly Media.

FEW, Stephen. 2013. Information Dashboard Design: Displaying Data for At-a-Glance Monitoring. 2nd ed. Burlingame: Analytics Press.

HARVEST. 2025. Track Your Hours & Projects with Time Tracking Software [online]. [cit. 2025-02-01]. Available at: https://www.getharvest.com/features

ISO 9241-210:2019. Ergonomics of Human-System Interaction – Part 210: Human-Centred Design for Interactive Systems. Geneva: International Organization for Standardization.

MAGUIRE, Martin. 2001. Methods to support human-centred design. International Journal of Human-Computer Studies. 55(4), pp. 587–634.

NIELSEN, Jakob. 1993. Usability Engineering. Boston: Academic Press.

NORMAN, Donald A. 2013. The Design of Everyday Things. Revised ed. New York: Basic Books.

ROTO, Virpi, Effie LAW, Arnold P. O. S. VERMEEREN and Jettie HOONHOUT. 2011. User Experience White Paper: Bringing Clarity to the Concept of User Experience. Helsinki: COST 294 MAUSE.

RUDD, Jim, Karen STERN and Steve ISENSEE. 1996. Low vs. high-fidelity prototyping debate. Interactions. 3(1), pp. 76–85.

VESSELOV, Sarrah and Taurie DAVIS. 2019. Building Design Systems: Unify User Experiences Through a Shared Design Language. Berkeley: Apress. ISBN 978-1-4842-4513-2.

Appendices A Interviews

Appendix A.1 Interview summary: Chief Technology Officer

Participant: CTO, Allodium Games

Format: Zoom, structured interview with notes

Conducted by: author

A. Technical overview

Q: What technology stack will ClickDown be built on?

A: The frontend will use Svelte, the backend Node.js with TypeScript, and the database PostgreSQL. UI components will follow the Shadcn-Svelte library.

Q: How will ClickUp API data be accessed and stored?

A: The system will use existing ClickUp API endpoints where possible. Data will be mirrored into a local PostgreSQL database rather than queried live, because the ClickUp API is too slow for real-time report generation. When generating a report, data is first downloaded and then mapped to tasks from the local database. The local database is kept in sync with ClickUp through webhooks and scheduled reconciliation processes.

Q: How will authentication and role-based access work?

A: Users will authenticate via ClickUp Single Sign-On (SSO). Role-based access will be determined by the user's ClickUp workspace membership level.

B. Problem definition

Q: Why build ClickDown instead of using ClickUp's native reporting?

A: ClickUp cannot generate the consolidated reports the team needs. Without a locally downloaded copy of the data, report generation is not feasible at the required speed. ClickUp also lacks native logging and backup, so maintaining a local mirror ensures data continuity. The primary need is a single source of truth for all time-tracked entries across the workspace. Additionally, ClickUp's native search does not support queries without diacritics, which creates practical issues for the team.

C. Design and user requirements

Q: What features are required for version 1?

A: Three core features were identified: a login screen with ClickDown branding and a ClickUp SSO authentication button; a time report view showing tracked hours per freelance worker broken down by client; and role-based access, where freelance workers can only view their own data while the accountant and management can view the full team report.

Q: Who are the primary users of the system?

A: Three user types were identified: freelance workers, who access the tool to review their own monthly hours; the accountant, who needs hours grouped by client for invoicing and cashflow purposes; and management, who need an operational overview of team activity.

Appendix A.2 Interview summary: Accountant

Participant: Accountant, Allodium Games

Format: Zoom, semi-structured interview with screen sharing

Conducted by: author

Current workflow: The accountant currently receives monthly CSV exports of time-tracked data from ClickUp, prepared manually by the CTO. Each export contains time entries per freelance worker including the client against which the time was logged and total hours worked. The data is used to proportionally allocate invoice amounts across client projects for the company's cashflow reporting, a process carried out manually in a separate Excel spreadsheet.

Key pain points identified: Prior to the current arrangement, the accountant received up to twenty individual reports per month, one per freelance worker, and had to consolidate them manually. The consolidated monthly report was requested for approximately six months before it was produced; in the interim, historical data had to be recalculated retrospectively.

Missing client assignments are a recurring issue. When a time entry has no associated client, the accountant must contact the relevant freelance worker individually to clarify the attribution. The current workflow provides no mechanism to surface these gaps proactively.

Requirements confirmed during interview: One consolidated monthly report showing hours per freelance worker broken down by client is the core requirement, replacing the previous multi-file approach. Visibility of missing or unassigned client entries is needed to enable proactive follow-up. The report must cover the full team rather than individual people in isolation. Financial data such as invoice amounts is managed separately and is out of scope for ClickDown; the tool is required to provide hours data only.

Appendix B Design system

https://www.figma.com/proto/3DunRaBFf0dN6IaMYiPhSh/ClickDown?node-id=11-2752&p=f&t=PJ0Rcsvnu7zKc14B-1&scaling=min-zoom&content-scaling=fixed&page-id=10%3A613

Appendix C High-fidelity prototype

https://www.figma.com/proto/3DunRaBFf0dN6IaMYiPhSh/Prototype?node-id=54-714&p=f&t=5YPW9cIepe5lfXne-0&scaling=min-zoom&content-scaling=fixed&page-id=54%3A158