CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

The Missing Save Feature in Firefox

A Comparative Study of Browser Workspaces

Master's thesis in Computer science and engineering

Emil Holmsten

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

The Missing Save Feature in Firefox

A Comparative Study of Browser Workspaces

Emil Holmsten

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

The Missing Save Feature in Firefox
A Comparative Study of Browser Workspaces
Emil Holmsten

Supervisor: Swen Gaudl, University of Gothenburg
Examiner: Staffan Björk, Chalmers University of Technology

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

The Missing Save Feature in Firefox
A Comparative Study of Browser Workspaces
Emil Holmsten
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

While word processors have long included a Save command to prevent data loss, most browsers still require users to keep windows open to avoid losing their open tabs and ongoing work. This makes it difficult to pause and resume browsing sessions over time. Without a save feature, pausing and later resuming a browsing session becomes difficult and often depends on workarounds.

Although save features are common in programs like word processors, browsers did not implement window-saving until Opera introduced workspaces in 2019 [1]. Firefox still lacks a native feature to save windows. This raises two questions: Is a built-in "Save Window" function essential? And which design choices best support users in resuming tasks across multiple browsing sessions?

To explore how different tools handle tab and task management, a comparative analysis was conducted using Firefox as the baseline. Microsoft Edge, Vivaldi, and the Firefox extension Simple Tab Groups were included to compare browsers that support saving and restoring windows with Firefox, which lacks key features in this area. A task analysis was used to break down the steps needed to save and restore windows in each tool. Next, semi-structured interviews were conducted, identifying eleven user needs.

Based on these insights, two prototypes were developed. They both add a save button for unsaved windows, use clear default names and icons, and explore two different approaches to restoring saved windows: in-place and in a new window. In a between-groups usability study, both prototypes outperformed Edge and Vivaldi in most tasks in ease of use, user confidence, and task efficiency.

The study concludes with design guidelines for native window-saving tools and recommendations for future research. The results show that integrated window-saving features can enhance ease of use and user confidence in specific tasks, and offer usability advantages over current solutions in browsers such as Edge and Vivaldi.

Keywords: Computer, science, computer science, user experience, usability, window management, browser, project, thesis.

Acknowledgements

I want to thank my supervisor, Swen Gaudl, for his support throughout the project, even when he was no longer required to assist me. I greatly appreciate it.
I also wish to thank the participants who took part in the study, whose input was vital to this research.

Emil Holmsten, Gothenburg, May 2025

Contents

List of Figures

List of Tables

1
Introduction

The ability to save work and temporarily close files is a basic expectation in modern computing. As noted in [2], the idea that users might not want to save their progress is rare enough to be disregarded in most cases. Ideally, saving happens automatically in the background without user intervention. Despite this standard, browsers have only recently begun to support saving open windows.

Web browsing often involves repeatedly returning to the same pages over extended periods, often weeks or even months [3]. However, many browsers do not provide a dedicated feature to save and resume the state of open windows. Modern browsers typically retain tabs and windows between sessions, but this is mainly intended for crash recovery and short-term continuity. Without dedicated saving features, users often rely on this limited persistence as a workaround, leaving browser windows open between sessions to avoid losing their tabs and ongoing work, even while working on unrelated tasks. This behaviour highlights a persistent inconsistency in software design: most applications support saving and resuming work, but browsers still lack this capability. This feature gap was first addressed in 2019 when the Opera browser introduced permanent window saving with its Workspaces feature [1].

Firefox is a widely used, open-source browser with robust extension support, but it lacks a native feature for saving and restoring windows. This makes it a natural baseline for evaluating whether such functionality is better delivered through integrated features or implemented via extensions.

Previous research has frequently used novel approaches to improve revisitation, task management and tab management [4], [5]. In contrast, this thesis will focus on evaluating and refining existing solutions to identify areas for improvement.

This study takes a hybrid approach that leans towards activity-centred design, with an emphasis on understanding the task in depth, while being guided by selected human-centred design principles where appropriate. The study will involve conducting interviews, studying previous research, and examining the current task and competitors' solutions. Based on this understanding, initial user requirements will be established. After a prototype has been developed and tested on users, a set of design recommendations will be made.

The rest of this thesis is organised as follows. Chapter 2 provides background information on multi-session web tasks, tabbed browsing challenges, and existing workspace solutions. Chapter 3 outlines the theoretical framework that informs the study. Chapter 4 describes the methodological approach, including the research methods and design process. Chapter 5 presents the thesis plan and timeline. Chapter 6 provides a detailed explanation of the execution and development process. Chapters 7 and 8 contain the competitor and task analyses, respectively. Chapter 9

describes common user workflows, while Chapter 10 summarises themes from user interviews and defines their needs. Chapter 11 reports on the comparative usability test, followed by an evaluation of the results in Chapter 12. Chapter 13 presents key insights, and Chapter 14 concludes the study by addressing the research questions and providing design recommendations. Finally, Chapters 15 and 16 reflect on recent developments and suggest directions for future work.
This thesis aims to address the following research questions:

- What do users need to manage browser tabs, windows and tasks effectively?
- Does Firefox support these needs?
- Can a Firefox extension meet these needs?
- How can window-management tools (workspaces) be improved?

2
Background

This chapter reviews relevant background on browsing behaviour, personal information management, and task organisation. It also introduces browser workspaces as a tool for improving long-term task management. Finally, it describes the theoretical frameworks and methods used throughout the study.

Browsing Fundamentals Modern web browsers provide a set of core elements that shape how users interact with the web.
A browser window is the main container for displaying web content. Users can open multiple windows to separate different contexts, such as work and personal activities.
Users can open multiple tabs within each window, each displaying a separate web page. Tabs are the primary way users multitask and keep several pages open simultaneously.
A browsing session refers to a continuous period of activity in the browser, but modern browsers often persist sessions across restarts. This means that open windows and tabs can be automatically restored, allowing short sessions to be extended into longer periods of use.
Users often return to previously visited pages. Tools like bookmarks and browser history support this, but they require deliberate use and are often underutilised.
Extensions are small browser programs that enhance the browsing experience by adding features such as ad blocking, password management, and organisational tools. They allow users to customise their browsers to suit their needs. Extensions use predefined interfaces (APIs) that interact with browser functions such as tabs, bookmarks, and more.
Finally, most browsers now allow syncing across devices, letting users carry their tabs, history, and bookmarks across phones, laptops, and other platforms.

Personal Information Management Personal Information Management (PIM) research shows that users often return to previously found material and face challenges managing their growing digital collections. Whittaker describes users as curators rather than just consumers, stating that recalling where and why something was saved implies cognitive burdens [6]. Users rely on informal strategies, such as leaving items in inboxes or piles, as well as more structured tools like folders and bookmarks. Jones identified various PIM practices, finding that users tend to favour low-effort, familiar methods unless more complex systems provide clear, practical benefits [7]. These papers suggest that the design of PIM tools can reduce mental

effort, but they must fit naturally into existing workflows and provide clear, practical benefits.

Multisession Web Tasks MacKay and Watters define multisession web tasks as goal-based tasks that require more than one browsing session to complete [3]. These tasks are carried out within the browser and typically involve activities such as online research, vacation planning, or information gathering that span multiple tabs and sessions. They found that users perform different types of multisession tasks, often consisting of multiple subtasks where half of the tasks lasted more than one week, 22% lasted 1-3 weeks, and 26% lasted even longer. They identified three main difficulties for users: keeping track of multiple multisession tasks, dealing with distractions, and restoring the task state when starting a new session.
Based on this feedback, they proposed that browsers should maintain a list of active multisession tasks, provide reminders to keep users focused and help users manage and organise their multisession tasks between sessions.

Web page Revisitation Web page revisitation refers to the act of returning to a previously accessed web page [8], and it is common that users revisit the same pages when browsing [9]. There are two types of revisitation: in-session revisitation, when users return to a website more than once during a single session, and post-session revisitation when users return to a page after the session has ended [5].
Traditional tools for post-session revisitation include bookmarks and browser history. These tools can be cumbersome to manage, and many bookmarks are never used [10], [11], [12].
Most modern browsers allow open tabs and windows to persist across sessions, enabling in-session revisitation and post-session revisitation through the same mechanism [5], [11], [13]. However, as the number of open tabs and windows grows, the system can become unwieldy and difficult to manage.

Conflicting Pressures in Tabbed Browsing A survey of 103 participants showed that many feel overwhelmed with around eight open tabs, an issue experienced weekly by 67% and daily by 17% [14].
Participants report that having too many tabs causes stress and makes it difficult to concentrate. When too many tabs are open, screen real estate also becomes an issue, making it difficult to navigate and have situational awareness. Too many tabs can also cause performance issues and drain the battery. Finally, there is both social and self-pressure to stay organised, causing some users to close tabs before sharing their screens with others.
Despite these issues, there are also many reasons to keep tabs open. Participants state that open tabs can serve as reminders, provide quick access to frequently used tabs, and offer an external mental model to help keep track of tasks. Participants also avoided closing tabs because they feared losing valuable information, especially since it can be difficult to judge both the current and future value of a tab. Finally, participants kept tabs they aspired to visit open, even knowing they would likely never visit.
The researchers identified three main areas of improvement:

- Browser interfaces need better ways to compartmentalise tabs by users' task contexts and structures so that users switch between different tasks and subtasks more easily.
- Browser interfaces should provide richer structures that can better reflect users' mental models, making it easier for them to orient themselves in the information space and resume prior tasks.
- Browser interfaces should help users better manage their attention by dynamically changing the appearance of tabs to reflect their current relevance, making it easier for users to focus on what is important.

Browser Workspaces Tabs let users open multiple web pages within a single browser window. This helps reduce clutter on the desktop taskbar, enables side-by-side viewing, and allows users to group related pages [13].
Still, browsers lack options for naming, saving, or easily restoring windows. As a result, users who rely on multiple windows to organise their work must manually keep track of them, often leaving many open to avoid losing progress. This is similar to using a text editor that does not allow saving, forcing users to keep every document open at all times.
Browser workspaces address this limitation by allowing users to name, save, close, and reopen entire windows along with their tabs. This helps users manage multiple ongoing tasks across sessions, reducing both mental effort and system resource usage.

History of Saving Windows Modern browsers have gradually introduced native workspace or tab-grouping features that help simplify multisession browsing. The first browser to introduce a workspace feature was Opera in 2019, followed by Safari in 2021, and Microsoft Edge and Vivaldi in 2023 [1], [15], [16], [17]. Chrome has not yet introduced a native workspace feature, but it has a tab-grouping feature that allows users to save these groups of tabs [18]. Firefox has not yet introduced native workspace or tab-grouping features, but it does have several extensions that provide similar functionality.

2.1 Related Work

This section describes previous research on how users manage web tasks, revisit pages, and organise browser content. Standard browser tools, such as tabs and history, often fail to meet the needs of users working on complex or long-term tasks. Several systems have been developed to improve task organisation, revisitation, and information sharing. The following subsections present key examples of these systems, along with their main features.

Contextinator - Project-based Management of Personal Information on the web Ahuja et al. developed Contextinator, a project-based management tool for researchers [4]. Their design was informed by earlier studies showing that users tend to group information by projects and require an efficient way to save and restore a project's state.

Projects in Contextinator are created automatically when the user opens a new window in the browser. Tabs added to the window are saved automatically. Projects can also contain to-do items, bookmarks, emails, and links to external applications. Their research found that users had their own individual ways of defining what counts as a project in this tool. For some, a project was a specific task. For others, it could be a broad topic, an ongoing activity, or just a group of tabs. These can be either short-term or long-term in nature. This shows that tools like this need to support many ways of organising work.
The researchers also found a difference between users who switched projects intentionally and those who switched more spontaneously. While they expected users to switch between and create new projects intentionally, they found that many users found themselves in the middle of a different project after having already opened a few tabs. This suggests that users should be able to create new projects easily from existing tabs and windows, but more generally, the system should be flexible enough to support disorganised workflows.

Sailboat - Task-based Browsing Kulkarni et al. introduced Sailboat, a Chrome extension that helps users organise web browsing by tasks [19]. Each task keeps its tabs, history, bookmarks, downloads, and archived pages in one place. Sailboat adds a collapsible dock on every page, allowing users to easily switch, create, or close tasks. It tracks the time spent on tasks and allows users to archive useful pages in a personal, searchable collection. In a study with 20 users over seven days, participants reported fewer distractions, improved tab management, and easier retrieval of past information. Time-tracking was also found helpful for maintaining focus.

TabFour - Task Grouping and Interruption Recovery Rajamanickam et al. developed TabFour, a prototype task-focused browsing interface designed to help users manage tasks, recover from interruptions, and share web content [20]. Users can group tabs into tasks, add annotations, and switch between tasks using a taskbar and side panels labelled "Active" and "Inactive." Tasks also store related bookmarks. A user study with eight participants reported improved organisation and ease of task switching and sharing.

These three tools show that users benefit from organising their browsing by tasks. Users found it easier to manage tabs, switch between tasks, and return to their work. The studies also show that task creation needs to be flexible, as users define and switch between tasks in various ways. Tools should support both planned and spontaneous workflows.

2.2 Theories

This section outlines the theories that will be discussed in the theory and methodology chapters.

2.2.1 Research through Design

According to [21], Research through Design is a type of design research that focuses on creating new knowledge through the design process. This approach differs from other research paradigms in that it uses design practice as a method of inquiry, with the creation and exploration of design artefacts serving as the primary means of investigating research questions.

Process In Research through Design, as described by [22], design researchers tackle wicked problems by ideating, iterating, and critiquing potential solutions to re-frame the problem to make the right thing; artefacts intended to transform the world from the current state to a preferred state. The result is a concrete problem framing and articulation of the preferred state, along with models, prototypes, products, and design process documentation.

Research Artefacts According to [22], research artefacts and design practice artefacts differ in two key ways. Research artefacts aim to produce knowledge for research and practice communities rather than to create commercially viable products. Research contributions should also be significant inventions that integrate theory, technology, user needs, and context rather than refining existing products. The artefacts represent theoretical and technical opportunities and can enhance knowledge transfer among research, practice, and education communities. They also create community discussions, allow researchers to analyse and search for similar approaches, and enable the emergence of design patterns.

Evaluation Design research interaction design research can be evaluated using four lenses: process, invention, relevance, and extensibility [22].

- Process: Whether the design research is described in detail to ensure the reproducibility of their methods and provide a rationale for selecting specific methods, ensuring that their work has been rigorously applied and chosen.
- Invention: Whether the design research is a significant invention. A thorough literature review is required to contextualise the work and highlight its contribution to the research community's knowledge and potential for significant advancements.
- Relevance: How well design research is framed within the real world and how clearly the researcher has articulated and argued for their preferred state.
- Extensibility: How well the design research is documented and described so that the community can utilise its knowledge, either by applying the process to future design problems or understanding and applying knowledge created by the resulting artefacts.

2.2.2 Human-Centered Design

Campese et al. [23] examine the various definitions and interpretations of Human-Centered Design (HCD) and User-Centered Design (UCD). The authors extensively reviewed literature and standards, identifying 21 different definitions of HCD/UCD.

They observed significant confusion around these terms, noting that they are often described in conflicting ways, such as an approach, a process, a philosophy, and a set of principles or methods.
Still, they found seven elements that define HCD:

User Involvement User involvement is essential for understanding user needs and preferences, thereby shaping the design process to be more user-centric. The emphasis is on making the user a central figure in the design process, either as a source of information or as an active participant in the design process.

User information and knowledge The paper highlights several types of user information critical to UCD, including user requirements, tasks, goals, needs, values, perceptions, and concerns. This information forms the foundation for understanding what the user wants and needs from the product or system.

Activities The authors describe activities in HCD/UCD as detailed, specific actions performed during the design process that help achieve the desired outcomes. The researchers recommend aligning these activities with the different phases in the design process. Activities are also, importantly, iterative. These activities vary greatly in level of abstraction but are typically focused on understanding users and context, generating requirements, designing and prototyping, and evaluating the results.

Methods and techniques The authors describe methods and techniques as being at the operational level of HCD. The best choice of methods and techniques depends on the specifics of the design process.

Goals The authors agree that HCD should be applied to achieve well-defined goals, such as improving the product currently being developed or the design process itself. Several authors refer to usability itself as a goal for HCD or a comparable statement such as developing usable products. Others dispute this, saying that usable products are the goal of any product design, not just HCD. The diverse range of goals suggests that authors' interpretations of what goals are for can vary significantly.

Principles Although only a few authors explicitly mention principles in their definitions, many authors include principles in their work. Here is a subjective summary of the mentioned principles:

- Focus on users and tasks early and continuously.
- Actively involve users in the design process.
- Incorporate all user experiences in design decisions.
- Use simple designs and early prototypes for early feedback.
- Iteratively design, test, and redesign.
- Integrate HCD practices deliberately.
- Adapt the process to specific user needs and project requirements.
- Use objective methods for data collection and analysis.

- Assemble multidisciplinary teams, including usability experts.
- Adopt a user-centred attitude.

Connection with the phases of the design process Various authors and standards agree that HCD should be a fundamental part of the design process. The connection between the design process and HCD is also mentioned in several design standards, though how exactly is not very clear. The paper states that further research on this subject is necessary.

The connections of the HCD/UCD elements The authors state that the foundation of HCD is *user involvement*, either directly or indirectly, to acquire *user information and knowledge.* They consider *user information and knowledge, activities,* as well as *methods and techniques* to be essential when applying HCI. They emphasise the importance of collecting and understanding information about the user to ensure that the design meets their expectations. According to the authors, defining *activities* ensures that the design process includes iteration and frequent user involvement. These are usually performed using *methods and techniques* that are chosen based on the project's specifics. *Goals* determine how to apply HCD in the project and the activities and methods to be performed. They state that *principles* determine a team's HCD culture; they are essentially guidelines for focusing on the user. Finally, the authors describe the *connection with [the phases of] the design process.* They say that HCD should be an integral part of the design process but not a substitution for a design process.

Definition of HCD They recommend using a definition of HCD as an approach to refer to the HCD field as a whole. They suggest the following definition based on the various definitions they have reviewed:
"Human Centered Design is an approach for *user involvement* comprising *principles* that guide establishment of *goals* and the accomplishment of *activities*, enabling *user information & knowledge* to be obtained and incorporated in the design process, using *methods and techniques* identified in the body of knowledge of this area."

2.2.3 Activity-Centered Design

In his article "Human-Centered Design Considered Harmful," Donald A. Norman provides a critical perspective on human-centred design (HCD) in the context of software products [24]. Norman states that HCD, which focuses on users' needs and abilities, has improved the usability and understandability of software. He also points out that the issue of software complexity remains despite this. While HCD aims to tailor products to specific user needs, many successful products were developed without explicit user studies or adherence to HCD methodologies, and many successful products are designed to be used by almost anyone. Norman suggests that while HCD has brought advancements, successful design can also emerge from methods not strictly rooted in HCD principles.
In contrast to HCD, which primarily focuses on the users' needs and abilities, Activity-Centred Design (ACD) emphasises understanding the activities for which a

device or product is intended. By designing to meet the requirements of the activity, it will be more understandable to users.
In HCD, listening to user feedback is a fundamental principle. However, Norman points out that strictly adhering to user requests can lead to overly complex and incoherent designs. ACD, on the other hand, maintains a focus on both the activity and the user. This approach helps create a cohesive design. In ACD, if a user suggestion does not align with the core activity, it can more easily be discarded by an authoritative designer with a clear vision for the product.
"Great design, I contend, comes from breaking the rules, by ignoring the generally accepted practices, by pushing forward with a clear concept of the result, no matter what. This ego-centric, vision directed design results in both great successes and great failures."
The underlying principle here is that great design sometimes necessitates stepping away from conventional practices and user-centric approaches. It involves a balance between listening to users and adhering to a strong, coherent vision that drives the design forward.

2.2.4 The Double Diamond

The Design Council's Double Diamond describes a design process that begins with divergent thinking to explore an issue, followed by convergent thinking to take action.

- Discover: Understand the users' problems.
- Define: Redefine the challenge based on the insight gathered.
- Develop: Explore various solutions to the problem.
- Deliver: Test solutions and iterate on what works.

They emphasise the importance of user involvement and iteration in the process. For example, making and testing early-stage ideas can be part of the discovery phase.

2.3 Methods

This section outlines the methods I will discuss in the methodology chapter.

2.3.1 Secondary Research

Several sources emphasise the importance of researching prior work early in the design process [2], [25], [26]. Secondary research can provide additional context, historical insights, and specific data that improve the understanding of the design challenge. They suggest this approach can be more resource-efficient than primary user studies. Cooper states that reviewing literature in the product domain can help develop questions and equip designers with the necessary knowledge and vocabulary.

The information should be up-to-date and current, and the source must be directly relevant to the research [27]. The author and publication should be recognised as trusted authorities on the subject, ensuring that the cited sources are accessible,

clear, and unbiased. For web sources, the URL and website layout should also reflect a level of trustworthiness.

2.3.2 Competitive Audit

Cooper [2] suggests examining current product versions and competitors early in the design process. He suggests doing this through an informal or expert review of current design and competitive product interfaces, comparing them against interaction and visual design principles.

2.3.3 Competitive Usability Evaluation

Competitive usability evaluation is a method for comparing a product to its competitors based on usability metrics, features, content, or design elements. It can be conducted as a competitive usability test, where users complete tasks in two or more competing designs, or as an expert review, where a usability expert compares the designs based on their experience to identify areas for improvement and potential design additions.
A competitive evaluation might evaluate 2 to 4 competitors with the best user experience or especially innovative design. The method is recommended as an initial research activity since it can help shape the project's design direction and determine the need for specific features.

2.3.4 Interviews

Sharp et al. [28] describe four types of interviews: open-ended, structured, semi-structured, and group interviews. An interviewer leads the first three and differs in how predetermined the questions asked during the interview are. The fourth type, a focus group, consists of a small group led by a facilitator. The purpose, questions, and interaction design activity determine the most appropriate interviewing approach. Cooper [2] suggests keeping the interview process to about an hour with about six interviewees per user role or type.

2.3.5 Task Analysis

A task analysis studies how users complete tasks to achieve their goals [29], [30]. It starts by gathering tasks and goals, followed by a review of the tasks. A task analysis is used to ensure that products and services are designed to support the users' goals effectively and appropriately [29].
According to [29], a combination of methods are typically used to learn about user goals and tasks, including:

- Contextual inquiry: The task analyst conducts a semi-structured interview with the user, then observes their work and interviews them about what they saw.
- Interviews using the critical incident technique: Users recall critical incidents, with interviewers asking follow-up questions to gather details.

- Record keeping: Users must keep track records or diary entries of their tasks over a set time period; tracking software can also be used.
- Activity sampling: Users are to track tasks, their duration, and frequency.
- Simulations: The task analyst follows the steps that a user would take when using the system.

[29] states that research should not solely rely on self-reported behaviour or simulations but also observe the user to capture important nuances or details. However, [30] suggests that if the design problem is simple and time is short, performing a Simulation could be enough.
After determining the goals and tasks, an analysis is performed, often resulting in a diagram. This diagram helps communicate the task flow to others and can be used to analyse attributes such as the overall number of tasks, their frequency, or the time required to complete them [29]. It can also be used to identify optimisation opportunities [30].

2.3.6 Thematic Analysis

Thematic analysis is a method for identifying, analysing, and reporting patterns within data. Thematic analysis can be conducted in two primary ways: inductively and deductively. Inductive analysis is a data-driven approach in which themes are closely linked to the data. The identified themes may have little relationship to the research question, indicating freedom from the researcher's theoretical interests. This approach avoids fitting the data into a pre-existing coding frame or the researcher's analytical preconceptions, which enhances objectivity and reliability. However, it is important to note that researchers cannot completely detach themselves from their theoretical and epistemological commitments.
On the other hand, theoretical thematic analysis is driven by the researcher's theoretical or analytic interest in the area. This form of thematic analysis, in which the researcher takes a more active role, provides a less detailed description of the data overall and a more detailed analysis of some aspects.
The choice between inductive and theoretical approaches is guided by the role of the research question in the coding process. Researchers can code for a specific research question, aligning with the more theoretical approach. Alternatively, the specific research question can evolve through the coding process, aligning with the inductive approach.
Thematic analysis is conducted in six phases. The process is not linear but recursive, and it evolves over time.

Phase 1: Familiarising yourself with your data The process of analysing data involves immersing oneself in the data by reading it repeatedly and actively, searching for meanings and patterns. To conduct an effective analysis, it is important to be familiar with all aspects of the data, whether for an overall or detailed analysis. Skipping this phase should be avoided as it provides the foundation for the rest of the analysis.
Transcription of verbal data, such as interviews, is a crucial step in the thematic analysis process. Despite its time-consuming nature, transcription can also help

familiarise oneself with the data and develop a thorough understanding of it. It is essential to maintain accuracy in transcripts, as punctuation can alter the meaning of the data. Transcription conventions vary, but the process is essential for effective analysis. Thematic analysis does not require as much detail in a transcript as conversation, discourse, or narrative analysis. The key is to retain the necessary information in a way that's true to its original nature.

Phase 2: Generating initial codes The process of coding involves producing initial codes from data, which identify a feature of the data that appears interesting to the analyst. These codes are part of the analysis, as they organise data into meaningful groups. However, the coded data differs from the broader themes, where the interpretative analysis occurs.
There are various ways of coding extracts, such as writing notes on texts, using highlighters or coloured pens to indicate potential patterns, or using Post-it notes to identify segments of data.
Important guidelines for this stage include coding for as many potential themes as possible, keeping a small amount of the surrounding data, and remembering that each data extract can be coded under several themes.
Also, no data set is without contradiction, and a good thematic "map" does not have to smooth out or ignore tensions and inconsistencies.

Phase 3: Searching for themes In the thematic analysis process, this phase involves sorting codes into potential themes and focusing on how different codes may combine to form an overarching theme. Initial codes can form main themes and sub-themes or be discarded. Unrelated codes may be grouped under a "miscellaneous theme" to temporarily categorise codes that do not fit into the main themes. Tables, mind maps, or writing code names and descriptions on a separate paper to organise them into theme piles are good methods for further exploration.

Phase 4: Reviewing themes After generating potential themes, it is time to refine them. It is essential to determine whether some themes should be merged or whether they are not themes at all due to insufficient or overly diverse data. Other themes may need to be divided into multiple ones. The themes should be clearly distinguishable from each other, and the data within them should make sense together.

Phase 5: Defining and naming themes Phase 5 involves further defining and refining the themes to be presented in the analysis. This involves identifying each theme's essence and determining what aspect of the data it captures. The next step is to write a detailed analysis of each theme, considering its story and fit within the broader narrative by examining the themes themselves and their relationships to one another. Sub-themes, which are themes within a theme, can provide structure to complex themes and demonstrate the hierarchy of meaning within the data.

Phase 6: Producing the report Phase 6 of a thematic analysis involves the final analysis and write-up of the report, which should tell the complex story of the

data to convince the reader of the merit and validity of the analysis. The write-up should provide a concise, coherent, logical, non-repetitive, and interesting account of the data's story. It should provide sufficient evidence of the themes within the data, using vivid examples or extracts that capture the essence of the point without unnecessary complexity. The extracts should be easily identifiable as an example of the issue. The write-up should also embed extracts within an analytic narrative that compellingly illustrates the story, making an argument about the research question.

2.3.7 Kruskal-Wallis Test

The Kruskal-Wallis test determines if different groups come from the same distribution. It tests the null hypothesis that there is no difference among the groups versus the alternative that at least one group is different from the others. This test is non-parametric, meaning it does not require the data to follow a normal distribution. Instead, it converts data into ranks and checks if the average rank is similar across the groups [31]. This method is especially useful for ordinal data or when the data does not meet the assumptions required for a one-way ANOVA [32].

2.3.8 Dunn's Post-hoc Test

Dunn's test is a follow-up method used after a significant result from a Kruskal-Wallis test. In other words, once a difference among the groups has been determined, Dunn's test identifies which specific groups differ from each other [31]. It is a non-parametric procedure for comparing pairs of groups, and it should only be used if the Kruskal-Wallis test finds an overall significant difference. This test is appropriate for three or more independent groups with ordinal or non-normally distributed continuous data.

2.3.9 Chi-squared Test of Independence

The Chi-squared Test of Independence checks if there is a significant relationship between two categorical variables [33]. For the test to work properly, the observations must be independent, and each cell should have an expected count of at least 5. A significant result means that the distribution of one variable is related to the other, while a non-significant result suggests that any differences are likely due to chance.

3
Theory

This chapter adapts general design theories to better align with the scope of the thesis by adapting the definition of Human-Centred Design (HCD) as presented by Campese et al. [23]. The adapted theory also incorporates aspects of Activity-Centred Design (ACD) [24]. This theory informs the methodological choices in chapter 4.

3.1 Adapting Human-Centred Design to the Thesis

According to research by Campese et al. Human-Centred Design (HCD) can be defined as an approach, process, philosophy, or set of principles or methods [23].
A loose interpretation of the definition found in section 2.2.2 is:

Human-centred design is an approach for *user involvement* where *principles* guide the establishment of *goals* and accomplishment of *activities.* During *activities*, *methods and techniques* are used to obtain and apply *user information and knowledge* in the design process.

The following section will discuss this definition and adapt it to the needs of this thesis. The section after that presents the adapted theory.

3.1.1 Discussion of HCD Theory

The definition above captures the broad scope of human-centred design as a field. However, this thesis requires a more specific interpretation suited to its goals and constraints. The original definition does not include secondary research, nor does it mention the iterative nature of design or the importance of selecting methods that match the specific needs of the project. The adapted theory also incorporates elements from Activity-Centred Design to address the practical aspects of this thesis. This approach helps analyse and improve concrete actions, such as saving, restoring, and switching between browser sessions. Based on these considerations, parts of the original definition are discussed below and adapted to fit the specifics of this thesis.

User Involvement Human-centred design focuses on the needs of the user [23]. The authors propose that this is achieved by involving users in the design process as sources of information or as active participants. While user involvement is a key

aspect of HCD, this definition may be too narrow since secondary research can also contribute to understanding user needs. As discussed in section 2.3.1, reviewing existing literature and prior work can provide relevant context, domain knowledge, and insights that help an initial understanding of users and their requirements. Therefore, this thesis adopts a broader interpretation that emphasises understanding user needs rather than limiting the process to only direct user involvement.

Principles The authors propose that design principles guide both the establishment of goals and the accomplishment of activities [23]. The role principles have in guiding activities is clear, but their impact on goal setting is less obvious. Seemingly, only one of the principles supports establishing goals, specifically *adopting a user-centred attitude*. However, overall, it seems more consistent to view the principles as primarily guiding the execution of activities, the selection of methods, and the application of techniques.

Goals Suppose design principles do not directly guide the establishment of goals. In that case, it may be more accurate to view them as guiding the choice and execution of activities in a way that supports the achievement of user-centred goals. This thesis's adapted theory will reflect the interpretation above.

Activities The original definition by [23] does not explicitly address the iterative nature of design activities. Including this aspect in the adapted definition is important, as iteration is widely acknowledged as a fundamental feature of the design processes. Design often involves repeated cycles of understanding, creating, and refining, with the flexibility to move between activities as required.

Methods and Techniques The definition of HCD by [23] does not clearly reflect the need to choose methods based on the specifics of the design challenge. For example, early-stage user interviews may be suitable in some situations, while in others, building a prototype for user testing might offer more useful feedback. The adapted theory for this thesis will emphasise the importance of selecting methods and techniques that align with the specific design challenge and constraints.

3.1.2 Adapted HCD Theory

Based on the arguments above, the following definition will be used in this thesis: Human Centred Design is an approach guided by *principles* for *understanding users* by iteratively performing *activities* using *methods and techniques* selected to achieve user-centred *goals*.

Goals This thesis aims to contribute to the understanding of user needs and task requirements, informing future design efforts to enhance the user experience of browser session management.

Principles The principles of HCD are intended to guide the design process. The following principles have been selected based on their relevance to this thesis:

- Focus on users and tasks early and continuously.
- Actively involve users in the design process.
- Incorporate all user experiences in design decisions.
- Use simple designs and early prototypes for early feedback.
- Iteratively design, test, and redesign.
- Integrate HCD practices deliberately.
- Adapt the process to specific user needs and project requirements.
- Use objective methods for data collection and analysis.
- Adopt a user-centred attitude.

User information and knowledge The knowledge acquired throughout the thesis is intended to improve the user experience. In early phases, existing research and technical analysis will be used to develop an initial understanding of user needs and task requirements. In later stages, this understanding will be refined through user feedback. While user goals and requirements remain the primary focus, other aspects mentioned by [23], such as user values, perceptions, and concerns, are not addressed in depth due to the thesis's scope and constraints.

Activities Organising design activities by project phase helps keep the process clear and focused. The Double Diamond framework breaks the design process into four phases: Discover, Define, Develop, and Deliver.
This framework aligns with the thesis because it offers a straightforward and adaptable structure at a suitable level of abstraction.

Methods and Techniques As discussed in section 3.1.1, methods should be selected to suit the thesis's specific context and user needs. The selected methods can be found in chapter 4.

3.2 Integrating Activity-Centred Design

While this thesis is primarily guided by HCD, certain elements of ACD are included to better address the thesis's practical aspects. ACD offers a task-oriented perspective that complements HCD's emphasis on user goals, preferences, and experiences by shifting attention to what users do and how they do it [24].
In browser session management, user efficiency and satisfaction are directly tied to the specific steps required for a user to save, restore, and navigate between sessions. Incorporating ACD helps the thesis focus on the details of these actions, including the steps users take and the problems they face. This task-oriented view complements HCD by grounding design decisions in how the system functions while aligning with core HCD principles.

3.3 Application of Theory

This thesis uses an adapted HCD framework with elements of ACD to guide method selection. Due to time and resource limits, early stages rely on existing research and task analysis. User involvement is introduced later to refine and validate the design. The Double Diamond model structures the process into four phases: Discover, Define, Develop, and Deliver. More details about the application of this framework can be found in chapter 4, which also outlines the specific methods used in each phase and a brief motivation of why they were selected.

4
Methodology

This chapter explains the design approach and the reasons behind the specific methods chosen for the thesis.

Approach Human-Centred Design (HCD) focuses on user needs, while Activity-Centred Design (ACD) is useful for analysing well-defined tasks. Since the project focuses on saving and resuming browser windows, a specific and well-defined activity, ACD is a good fit. At the same time, HCD ensures that the user is at the centre of the design process. Because of this, the project uses a mix of HCD and ACD.

Methods The methods in this project were chosen based on the theoretical framework presented in chapter 3. The project uses methods that collect user input, such as interviews, thematic analysis, and usability testing, which are typical in HCD. The more task-focused methods, such as task analysis and competitor analysis, are derived from ACD. By using methods from both frameworks, the project can focus on both user needs and examine how users complete key tasks to improve the user experience.

Activities The process follows the four stages of the Double Diamond model: Discover, Define, Develop, and Deliver. These phases help organise the project while staying flexible, as each stage can be revisited multiple times.

4.1 Motivations of methods

Below are the motivations for each method choice in the project, organised by the four phases of the Double Diamond.

4.1.1 Methods used in the Discovery phase

Secondary Research Secondary research was employed to gain an initial understanding of user needs and the broader problem, informing the development of interview questions and providing a background context for the project as a whole.

Competitor Analysis Competitor analysis was chosen to study different implementations of session management by compiling design knowledge which is already available through existing design artefacts. It helped identify common features, design patterns, and potential issues to avoid.

Task Analysis Task analysis was used to break down how users save and resume windows. It was selected to help compare workflows between browsers and identify which are more or less effective. By documenting the process in this way, it can more easily be communicated and analysed.

Interviews User interviews were chosen to gain direct insight into user habits and preferences. This method was used to confirm or challenge early assumptions and provide input for the design process.

4.1.2 Methods used in the Defining phase

Thematic Analysis Thematic analysis was selected to analyse qualitative data from interviews to identify user concerns, behaviours, and preferences. This analysis can inform design decisions throughout the project.

User Requirements User requirements were chosen to capture user needs and expectations in a structured format. These can be used in the design process and as a summative result on their own.

4.1.3 Methods used in the Development phase

Prototyping Prototyping was chosen to simulate user interactions with the relevant design solutions in a simple format. It allowed early testing and feedback without building a full product, focusing on important interactions while avoiding distractions from non-essential details during testing.

Comparative Usability Test In a comparative usability test, users evaluate different design solutions against one another. It was selected to evaluate the prototype against existing solutions by collecting user feedback to help assess the user experience and find usability issues.

4.1.4 Methods used in the Delivery phase

Evaluation A quantitative evaluation was chosen to measure whether the prototype improves the user experience compared to existing browsers in metrics relevant to the project.

Design Recommendation Design recommendations were chosen as a final step to guide future development and provide direction for similar projects.

5
Planning

The following chapter is dedicated to explaining the initial project plan and any updates to the plan that were required during the project. Figure 5.1 shows the initial time plan in the form of a Gantt chart, depicting a breakdown of the project's total duration of 20 weeks.

5.1 Writing the Thesis

The first four weeks of the project will be primarily dedicated to planning the report and conducting literature research. The planning report is a first draft of the introduction, background, theory, methodology and planning.
When the planning report is completed, the writing focus will shift slightly while continuing to work continuously to refine the thesis and expand the background. This will also involve writing about the project process, as well as the results, analysis, discussion, and conclusion. At the end of the process, the report will be finalised, and the formal steps required to complete the thesis will be taken, such as submitting, presenting, and likely acting as the opponent for another master's thesis.

5.2 Main Project

The main project will be conducted in four stages based on the Double Diamond process: discover, define, develop and deliver. These stages are flexible and can be returned to as needed during the project.

Discover During the discovery phase, relevant papers will be searched for and reviewed to provide context for the research and establish a foundation for the interviews and competitor analysis. Furthermore, additional interviews and user testing will be conducted.

Define During this stage, the initial user requirements will be defined based on the insights gathered in the previous stage. Following the first round of development and user testing, these user requirements will be updated to reflect any new information obtained.

Develop This stage of the project is dedicated to converting the user requirements into potential solutions and then implementing these ideas as prototypes to be tested

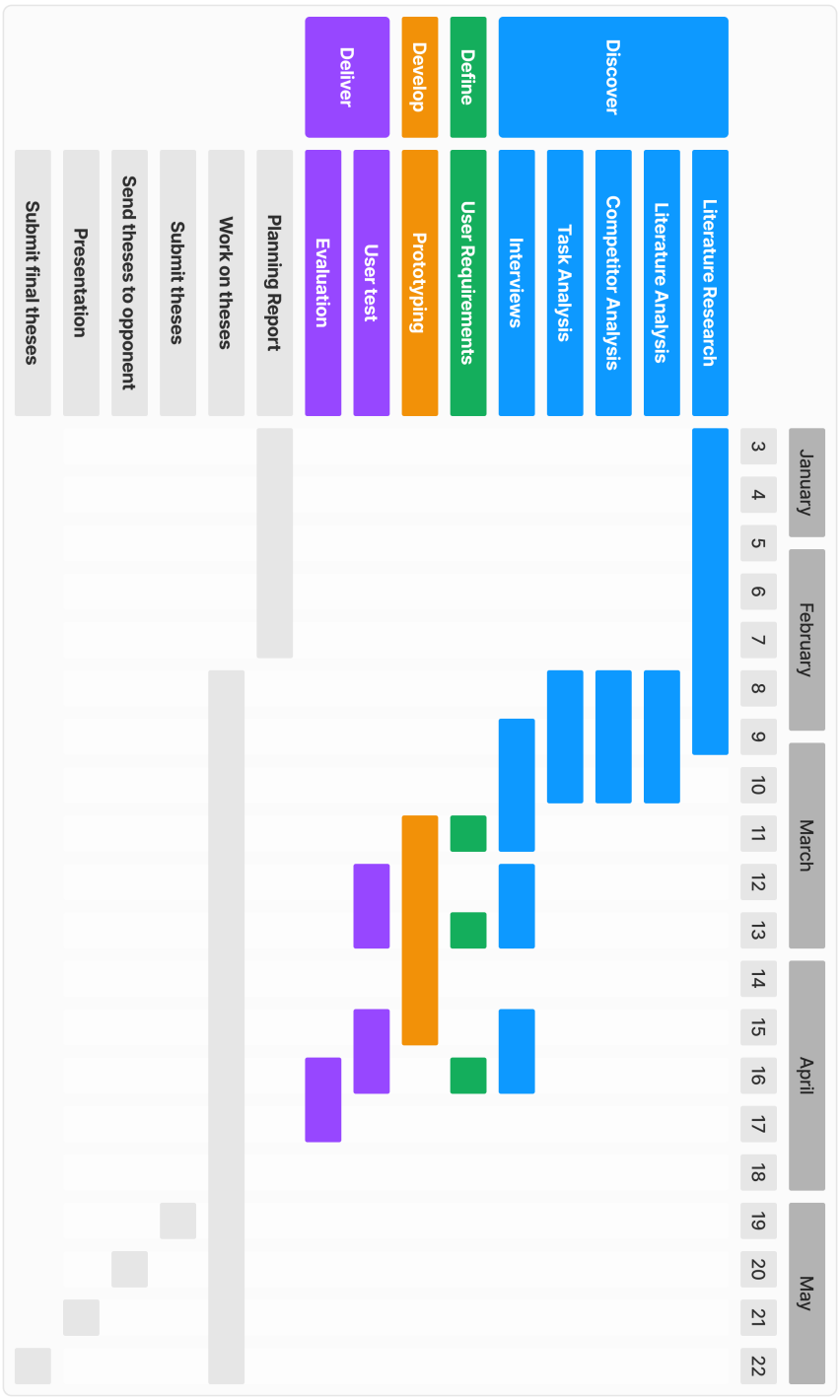


Figure 5.1: Gantt chart showing the initial time plan.

on users.

Deliver The final stage of the project will involve user tests to identify which solutions are effective and which are not. Following the completion of the last user test, the project results will be evaluated, and recommendations for future work will be proposed.

5.3 Project Delays and Continuation

Although the initial plan covered 20 weeks, the project did not proceed according to schedule. The practical components of the work, including the literature review, interviews, analysis, and user testing, were completed in rough alignment with the original plan for 2024, approximately during week 24. After that, the project was paused for approximately 8 months due to other studies and obligations. No significant work was carried out during this period. The thesis was resumed in spring 2025, with a focus on writing, analysis, and reflection.

6
Execution and Process

This chapter describes the practical work that was carried out to develop and evaluate the window-saving tool. It details the methods used to gather background information and user insights, starting with online secondary research and a structured competitor analysis. The chapter explains how tasks, workflows, and user needs were identified through direct observations, interviews, and task analysis. Based on these findings, several prototypes were developed and iteratively tested. Each section outlines the procedures and considerations taken to ensure that the design decisions were informed by real user data, laying a solid foundation for the final evaluation and the resulting design recommendations.

6.1 Secondary Research

The secondary research drew from various sources, including academic papers, browser documentation, browser forums, news articles, and blog posts. Some of these are not directly cited in the thesis but still contributed to the overall understanding of the topic and shaped the research. The research covered three main areas: user behaviour, design theory, and research methodology. For user behaviour, key studies included work by Whittaker and Jones on personal information management, MacKay and Watters on multisession web tasks, and papers on revisiting behaviour and tab overload. These sources helped identify key user problems, such as losing track of tabs and tasks and supported decisions throughout the design and research process. Studies of systems like Contextinator, Sailboat, and TabFour provided practical insights into how users manage tabs and sessions. Sources for design theory included models like the Double Diamond, Human-Centred Design, and Activity-Centred Design, which helped structure the overall design approach. Methodological sources included papers on thematic analysis, task analysis, interviews, and usability testing, which guided the application of these methods. This part of the work began in week 3 of 2024 and continued until week 10.

6.2 Competitor Analysis

A competitor analysis was conducted to identify best practices and understand the important features related to saving windows in major browsers and Firefox extensions. The results of this analysis can be found in chapter 7. The work started in week 8 of 2024 and lasted until week 11.

Candidate Selection Around ten tools were reviewed, and three were selected for closer analysis based on manual testing. Each tool was evaluated first for whether it included window-saving features, second for how usable those features appeared, and sometimes for other reasons. For example, Chrome's profile system offers similar functionality but does not reliably preserve window states. Safari was excluded due to its restriction to Apple devices. Opera includes a workspace-like interface, but it does not support saving windows and differs significantly in user experience from typical workspace designs and was excluded. Workspace extensions, such as Workona, were also considered as alternatives to Simple Tab Groups (STG). Workona specifically was excluded due to its complexity and deviation from other workspace tools. Instead, STG was selected to represent Firefox workspaces using extensions. Since Firefox lacks a native workspace feature, STG provided a comparable experience to the native workspace tools in other browsers. The final tools selected for the competitor analysis were Edge, Vivaldi, Firefox, and the STG extension for Firefox.

Method of Analysis Each of the selected competitors was reviewed using direct observation. They were downloaded and tested manually to explore their window-saving functionality. The process involved trying out relevant features and assessing their ease of use and overall design. While no formal framework was applied, the analysis focused on typical actions such as saving, restoring, and switching between workspaces. The tools were revisited later to confirm impressions or recheck unclear behaviours. Full descriptions of each tool and its features can be found in chapter 7. Although this method provided useful insight into core workflows, a lot of functionality related to workspaces in each browser was not described in detail in the analysis to manage the scope of the thesis.

6.3 Task Analysis

The task analysis was conducted to provide a more structured and objective comparison between the tools and complement the earlier subjective observations. The tools analysed included Firefox (both with and without the Simple Tab Groups extension), Edge, Vivaldi, and two experimental prototypes. The experimental prototypes were developed in a later stage of the thesis for the evaluation and were added to the task analysis later in the process. These prototypes are described in section 6.9.
Each step of each workflow was assessed in terms of click count and mental work. Click count included every physical user interaction, such as clicking a button or a menu item. Mental work refers to tasks that require mental effort to complete even after some experience with the tool. What counted as mental work and what did not was decided subjectively after repeated use and analysis of the tools, based on how much cognitive effort an experienced user would likely need. For instance, navigating a context menu with several options was considered mental work, whereas re-finding a button in a known location was not.
All tools were revisited many times to confirm behaviours or clarify specific task breakdowns. The work started in week 8 of 2024 and continued until week 11,

simultaneously with the competitor analysis and workflow breakdown. The results of the task analysis can be found in chapter 8.

6.4 Workflow Breakdown

To formalise the discussion around how users manage tasks in browsers, the workflow breakdown defines an exhaustive set of workflows using windows as the sole method of tab organisation. The model is grounded in real-world user behaviour based on interview findings that describe the use of multiple monitors, multiple devices, and varied task management strategies. Workflows were categorised using data modelling terminology: one task in one window (one-to-one), one task across several windows (one-to-many), multiple tasks in a single window (many-to-one), and multiple tasks across multiple windows (many-to-many). These categories aim to cover all possible ways users might structure their browsing activity. A tree diagram is included to illustrate the relationships and confirm the completeness of the breakdown, see figure 9.1. The framework highlights the diversity of real usage patterns and supports later discussions about designing for flexibility in window-based task management. The results from the breakdown can be found in chapter 9. This work started in week 8 of 2024 and lasted until week 11, simultaneously as the competitor analysis and task analysis.

6.5 Interviews

Five participants participated in a series of loosely structured interviews to explore their browser habits, identify common pain points, and confirm earlier research findings. The interviews and thematic analysis were conducted between the weeks 10 and 16 of 2024. These results can be found in chapter 10. The complete interview transcripts and coded data are included in the appendices A and B.

Participant Selection Participants were recruited through convenience sampling, mainly due to time constraints. All five had university-level education, with backgrounds in interaction design, economics, and geology. They were between 25 and 33 years old, comprising three males and two females. All participants used the web multiple times daily and described themselves as highly proficient browser users. Their experience with browser workspace and tab management features varied.

Interview Protocol Before the interviews, participants signed a consent form that explained the study's purpose, data use, and their right to withdraw from the study, as well as the study's commitment to maintaining participant anonymity. The process followed Swedish data protection guidelines. The interviews focused on browser use, particularly how participants managed tabs, windows, and tasks. Initial questions covered the number of tabs and windows typically open, their use of tab groups, multitasking habits, and how participants managed closing and saving windows. The questions evolved during the process, allowing later sessions to explore relevant topics in more detail. Depending on participant preference, interviews were

held in person or via video call. Each session was audio-recorded and lasted between 14 and 35 minutes.

Transcription and Analysis Recordings were transcribed using AI software and manually edited to correct any errors. The transcripts were analysed using thematic analysis in NVivo [34]. They were coded, reviewed, and grouped into themes as part of this process. Full transcripts are provided in appendix B, and coded data in appendix A. The resulting user needs are described in section 10.1.

6.6 Identification of User Needs

A list of user needs was created based on the results of secondary research, competitor analysis, task analysis, and interviews. This can be considered a summary of the findings from previous research in preparation for the usability tests. This list was created after the completion of the interviews and thematic analysis in week 17. The resulting user needs are described in chapter 10.

6.7 Comparative Usability Test

A comparative usability test was conducted to improve upon the designs, the same 5 participants from section 6.5 were called upon again to test three mockups of the window-saving interface.
The initial user test had three main objectives:

- Which method of switching between windows is most natural to the user?
- Which method of saving a window is most natural to the user?
- What are the strengths and weaknesses of the competitor's designs?

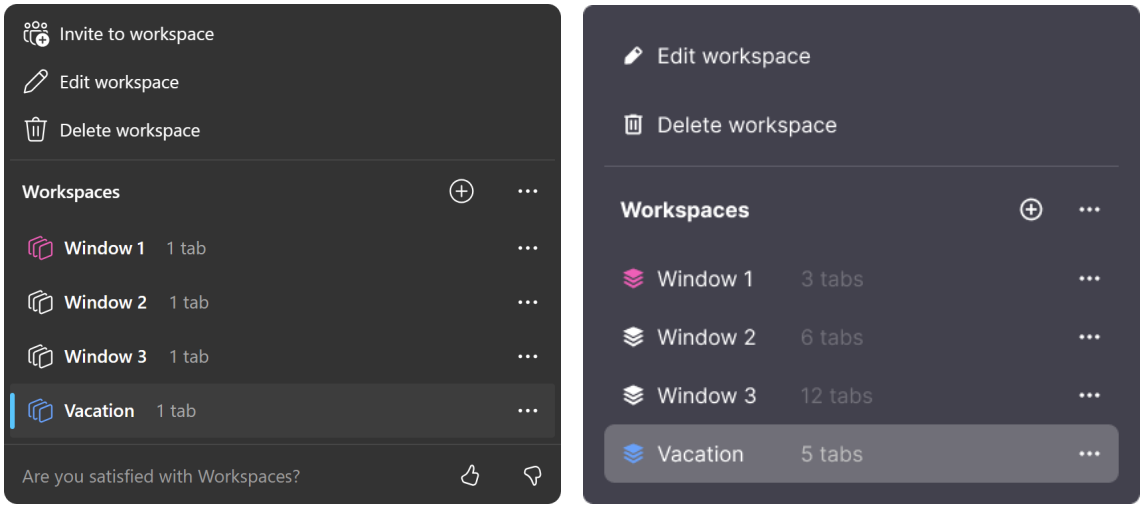
The comparative usability test was created and conducted from weeks 17 to 19. The results of the comparative usability test can be found in chapter 11.

6.7.1 Mockup Development

The three mockups were quite thoroughly implemented using Figma, allowing the user to switch between multiple windows, save and create multiple windows and rename them. It was essential to implement the mockups thoroughly to enable users to immerse themselves in and evaluate their experience with the window-saving functionality, including the specifics of opening and switching between windows, which work differently in each mockup. The result was three quite interactive mockups, which were deceptively difficult to implement due to the numerous connections required to simulate the various possible states resulting from the user being able to open and close windows and have different windows open in different positions.

6.7.2 Design Variants

The study compared the usability of three mockups. Two were closely based on Microsoft Edge's and Vivaldi's implementation of their window-saving interface, while



(a) Original design of the Edge window-saving interface.

(b) Mockup of the Edge window-saving interface.

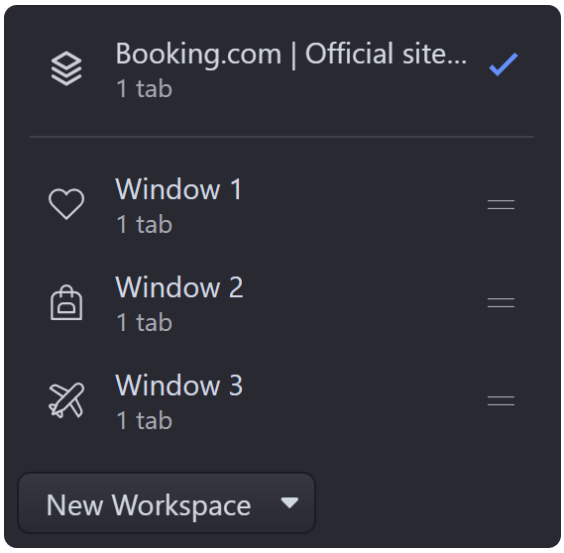
Figure 6.1: Comparison of the original and mockup Edge window-saving interfaces.

the third explored some alternative design choices. All three mockups were created following Firefox's visual design guidelines to ensure that participants' feedback primarily focused on the experience rather than visual differences. It also helped to hide which variant was designed by the competitors versus which one was designed for the study. The changes included the colour scheme, typography, and a few more heavy-handed changes, like removing the extra selection markers that both Edge and Vivaldi had to show which window was currently selected. While the final changes may be excessive, the additional visual indicators were determined to offer minimal advantages and could divert attention from the ongoing user test.

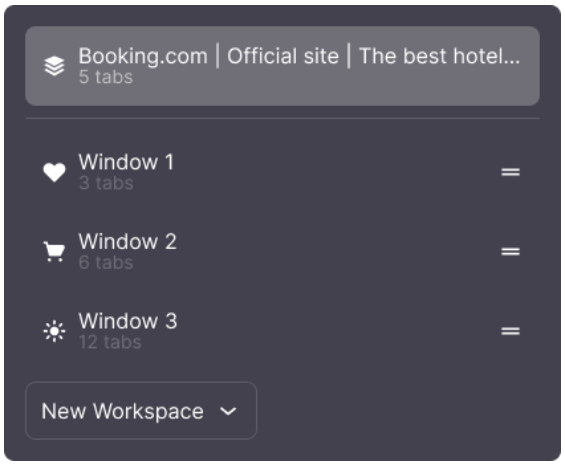
Edge Mockup Comparing figure 6.1 shows most differences between the two variants.

- Feedback Mechanism: The original design includes a feedback mechanism at the bottom, asking users to rate their level of satisfaction with the feature. This was removed from the remake.
- Workspace Management Options: The original design includes an "Invite to workspace" option, which is absent in the remake since the study does not investigate the possibility of collaboration options. The other options, "Edit workspace" and "Delete workspace," are present in both versions. These options are also available behind the triple-dot menu on the right side of each list item, and the "Invite to workspace" option is removed there as well.
- Selected Item Highlight: The original design included a small vertical line on the left side of the selected list item. The remade version had a stronger highlight on the selected list item instead.

Vivaldi Mockup Comparing figure 6.2 shows most differences between the two variants.



(a) Original design of the Vivaldi window-saving interface.



(b) Mockup of the Vivaldi window-saving interface.

Figure 6.2: Comparison of the original and mockup Vivaldi window-saving interfaces.

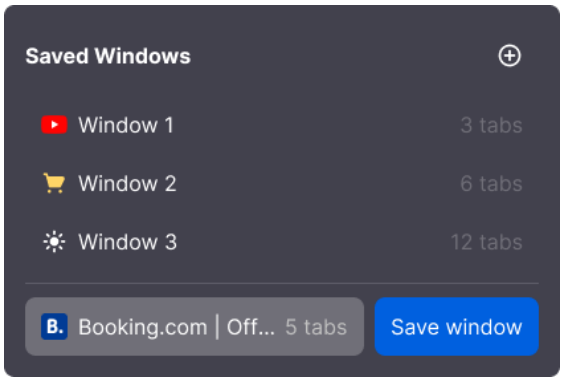


Figure 6.3: The original design for the exploratory window-saving interface.

- Selected Item Highlight: The original design included a small blue checkmark on the right side of the selected list item. The remade version had a stronger highlight on the selected list item instead.

Exploratory Mockup Comparing figure 6.1, figure 6.2 and figure 6.3 shows most of the differences between the three final variants in use during the first user test.

- Unsaved Window List Item: The exploratory version included the unsaved window feature from Vivaldi. However, it was moved down to be next to the button used to save it. Another option could have been to move the save button to the original position of the unsaved window list item.
- Save Window Button: Instead of hiding it behind a triple-dot menu like in Edge or behind a small drop-down arrow like in Vivaldi, the save window function was presented prominently next to the unsaved window, something that neither of the two competitors has opted for.

- New Workspace Button: The exploratory version uses Edge's more minimal design for creating new workspaces, which slightly deemphasises the action compared to Vivaldi.
- Triple-dot Menu: The menu item to the right on each list item was taken from Edge and implemented in the exploratory design. This is in contrast to Vivaldi, which does not include this menu at all and instead expects users to right-click to access additional options. However, the triple-dot menu is only visible when selecting or hovering over a list item.
- Current Website Title as Default Name: The exploratory version takes inspiration from the Vivaldi unsaved window list item, which is titled using the title of the last open website. However, the exploratory version goes one step further, allowing the user to keep the default window title after saving, inspired by how web browser windows are generally named in the operating system taskbar: by the title of the last visited web page, as shown in figure 7.1. This title changes if the user navigates to a different website within the window.
- Current Website Favicon as Default Icon: The exploratory version explored an alternative approach to using the current website favicon as the default icon. Instead of not allowing the user to change only the colour and not the icon, like Edge, or selecting a random icon that can later be changed, like Vivaldi, the exploratory version also takes inspiration from the taskbar. Similarly to how a window in the taskbar is named based on the current website the user is visiting, the list item in the exploratory version uses the favicon of whatever website was last visited. The user can choose a permanent favicon from any of the currently open websites in the window, allowing them to lock the icon to something they consider relevant.
- Allow both Icon and Colour change: Unlike competitors, which only allow either colour or icon, the exploratory version allows both to be changed and allows the user to use a website favicon instead of an icon.

6.7.3 Issues

Below is a list of potential issues that could have been done differently

- Neither Edge nor Vivaldi had a name attached to the icon in the top left; this feature is currently not possible to implement in Firefox either way, but testing it might have still been preferable.
- Naming in Vivaldi was not implemented in the prototype; instead of replacing the window name with an editable text field, the name was replaced instantly when the user saved or created the workspace. This means there is no test data for this implementation, an implementation which might be the best approach.
- Right-clicking to open the context menu was not implemented in Vivaldi. This was not a significant issue as the participants could be informed of this during the test if they tried to take this action.
- Both Edge and Vivaldi mockups could have been improved compared to the original designs to get more valuable results about how usable these specific designs are without these issues. The buttons that allow users to save their current window are hidden in both versions. On Edge, the user must click the

triple-dot menu to the right of the plus icon, and this could have been changed to a different icon to make it clearer to the user. On Vivaldi, the drop-down arrow to the right of "New Workspace" must be clicked to save the current window. This could be replaced with two separate buttons to improve the design and achieve more useful results.
- A fourth mockup implementing a different window opening strategy could have been tested. While Edge, Vivaldi and the experimental variant all had different strategies for opening windows, no variant tested Edge's approach but included the option to open in the same window. This variant would have exhausted the possible design space for this particular feature, and it is likely an improvement over the current Edge functionality.

6.7.4 Usability Test Protocol

Each participant was informed of and provided consent for the study's data collection in accordance with data protection regulations. Participants then tested each mockup sequentially. They sat in front of a computer and were presented with scenarios and tasks that guided them through the experience. The scenario was the same for the three mockups, and the tasks were as similar as possible for each variant. The users were encouraged to think out loud both before and during the test and were also asked follow-up questions where appropriate. The tests were conducted in undisturbed environments to minimise external variables, and each session was recorded to prepare for analysis.
To reduce bias, the mockups were presented in a semi-random order. The order was randomised, but if the randomisation resulted in a sequence where a mockup appeared first more than two times, that particular sequence was not used. Instead, another random order was generated. This approach was designed to minimise any bias resulting from the order in which the mockups were presented. The order of the fifth user test was generated entirely randomly.

6.7.5 Thematic Analysis Process

Thematic analysis was also selected as the analysis method of choice for the comparative user tests. These user tests were not transcribed because of the project's scope and time constraints. Instead, a more direct approach was used, where user feedback and observations from the recorded sessions were documented and analysed and then grouped into themes. This method was more efficient and provided a more natural usability analysis while sacrificing some academic merit since the results of this user study were intended to be primarily formative rather than summative.

6.8 Rapid Prototyping

Based on the insights gained from the comparative user tests, a single prototype was developed. This prototype was used in a rapid prototyping phase, where minor adjustments were made between user tests to implement and evaluate design decisions rapidly. This approach was practical, and changes were made where they seemed

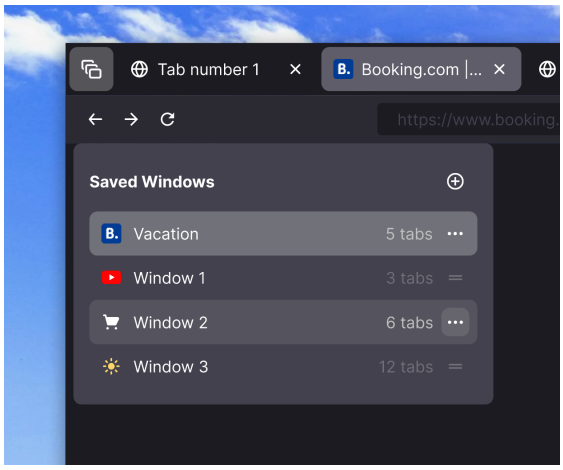


Figure 6.4: The original design for Vivaldi window-saving interface.

reasonable in response to direct feedback from users. This work was conducted after the comparative user tests in week 19 of 2024.

Prototype Development A prototype was developed based on feedback from the comparative usability test. Additionally, a few extra ideas were tested during the rapid prototyping phase. An example of additional features being tested can be seen in figure 6.4.

Usability Test Protocol The test procedure for the rapid prototyping phase was similar to comparative user tests. Following data protection regulations, each participant was informed of and provided consent for the study's objectives and procedures. Then, participants were seated at a computer, where they were presented with a scenario and tasks designed to guide them through the features of the updated prototype. Although only one prototype was tested during this phase, the tasks remained consistent with the previous study.
The users were encouraged to think out loud both before and during the test and were also asked follow-up questions where appropriate. The tests were conducted in undisturbed environments to minimise external variables, and each session was recorded to prepare for analysis.

Thematic Analysis Process Thematic analysis was once again the method of choice for the rapid prototyping user tests. Same as before, these sessions were not transcribed due to time constraints. Instead, observations and user feedback from recorded sessions were directly noted and analysed.

6.9 Evaluation

The summative evaluation of two exploratory prototypes, compared to mockups based on Edge and Vivaldi, will provide a quantitative measure of their performance to assess usability, performance, and user preferences for each design variant through

a structured set of tasks and questions. Participants completed these tasks using one randomly assigned mockup, and their interactions, success rates, and feedback were collected and analysed. A total of 59 participants were recruited for the study, with 14 participants testing Edge and 15 testing the others. The study was conducted online, and participants were recruited through various channels, including Reddit and a class group chat.
The evaluation process, from the initial design of the prototypes and tasks until the programming of the information visualisation and data analysis, took place between weeks 18 and 24 of 2024. The results of the evaluation are presented in chapter 12.

Participant Selection The study was posted on various browser-related forums on Reddit. The link was also shared in a class group chat. The link automatically and randomly distributed participants across the different study groups.
An imbalance in participant numbers persisted among the study groups. To address this, convenience sampling was employed to recruit the remaining participants. These participants were randomly assigned to groups by determining the number required for each group and then randomly assigning participants to each vacant spot from a list of willing participants.

Mockup Development Similarly to the previous user tests, mockups were developed in Figma. For the evaluation, it was not necessary to create complete, interconnected mockups of the tools, as the tests were structured with fixed start and end points. However, to ensure that all interactions happened correctly in each mockup, they were still implemented so that they were completely interconnected and fully functioning. Afterwards, they were broken down into smaller pieces to match the requirements of each task during the evaluation.

During the implementation of the mockups, a significant issue became apparent. Figma does not support right-click interactions, which are necessary for Vivaldi to function correctly. Based on the results from the rapid prototyping stage, the right-click option should also be a feature in the experimental prototypes. To solve this problem, a popup was added to the mockups. The popup appears when the user clicks on something, allowing them to select between the right-click and left-click actions.

The popup introduced bias since the user will be influenced by it, for example, by testing the right-click action when they had not intended to do so. A test mockup was implemented where every click opened the popup and another where the popup appeared only when right-clicking was an option. The idea was to avoid informing the user when a right-click action was possible, although they would still be constantly reminded that right-clicking was an option.

However, the pilot study revealed that the resulting mockup was not intuitive to use, as every action resulted in a jarring popup. Additionally, this mockup was extremely difficult to implement, as the already substantial number of Figma interaction connections was multiplied by this approach. This resulted in a mockup with

several hundred connections, making it very difficult to work with and also making it challenging to ensure that the mockup was correctly implemented.

In the end, the final mockups were implemented with the popup only appearing where right-clicking was an option. Hopefully, this did not massively affect the results, but there is no easy way to account for it properly. To minimise the impact of the problem, participants in the test were asked to select the option they initially intended when clicking and avoid choosing an unintended option, as this would skew the test results.

Developing the Evaluation Creating the evaluation took a significant amount of time and effort, and encountered several challenges along the way. The original plan was to conduct a comparative evaluation where the participants tested each of the four mockups in a random order. The approach was eventually changed to testing only one prototype per participant and randomising which of the four prototypes were tested due to technical limitations and to reduce the study's length. This is discussed in more detail further down.

The evaluation started with a set of questions to be answered, which then guided the design of the evaluation. A snapshot of the development for the tasks can be seen in figure 6.5. Five pilot studies were conducted with three different participants selected for convenience. In this phase, several changes were made to the evaluation, including modifying the questions to be answered and reducing the number of tasks from nine to five.

To track the results, an online tool, Pathway [35], was used, which could create tasks from Figma prototypes and include questions that participants could answer between tasks. Pathway enables tracking of success rates, mis-click rates, time spent on tasks, and other metrics.

Considerable time was spent figuring out how to set up a randomised order for the prototypes and creating the test in general. Late in the process, Pathway was found to be unable to develop the test as planned. Because of this, another tool, called Marvel [36], was chosen, and the study was recreated using this tool. Because previous pilot studies had shown that the evaluation took too long to complete, and because randomising the order in which each prototype was presented to each user could not be achieved in Pathway or Marvel, the original method of testing four prototypes in succession was replaced by having participants test only one of the four prototypes chosen randomly.

At this point, four more pilot studies were conducted using four new participants hired through the online tool used to create the evaluation. The difference between pilot studies with participants who were directly invited to participate and those who were paid online participants was substantial, most likely due to the latter skimming over questions and selecting answers randomly when frustrated or bored. The last pilot studies provided valuable information, which led to the implementa-

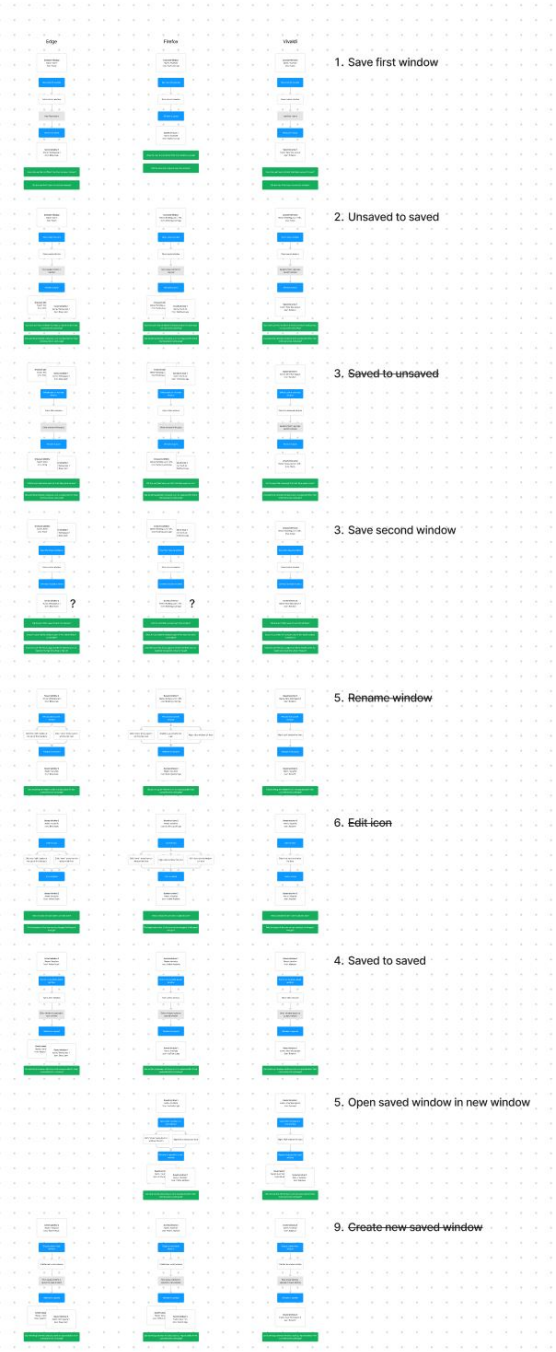


Figure 6.5: A work-in-progress picture of the process of designing the tasks.

tion of several changes to the questions and tests in general. These changes included:

- An additional arrow and guiding text were added to help users locate the tool on the first task, as finding the tool was not intended to be measured and would only add noise to the data.
- Reducing the number of questions about switching windows from three to one since users did not understand the difference between the questions at a glance.
- Adding more images to the questions to remind participants of the task they completed, for example, using before-after pictures of the task they had just completed.
- Reformulating some questions to get more reasonable responses since participants selected options that did not make sense.
- Editing the standardised set of questions at the end to be more understandable, users need to look up the meaning of some words; the following questions were changed while trying to retain their meanings:
  - Original: I found the various functions in this system were well integrated.
    Edited: I found the various functions in this system to work well together.
  - Original: I found the system unnecessarily complex.
    Edited: I found the system needlessly complicated.
  - Original: I found the system very cumbersome to use.
    Edited: I found the system very inefficient to use.

The complete set of questions and tasks can be found in 6.9.3.

The study was then posted on Reddit, specifically on forums closely related to browsers. This introduces bias in participant selection towards expert browser users but likely increases the study's response rate. The response rate was initially very low, and several potential participants were suspicious of the link that was posted, the author behind it (me), and the type of data being gathered, even though it was pretty explicit. Several changes were made to address these issues, and the response rate appeared to improve. Additionally, a technical issue prevented at least one user from completing the study and attempts to replicate and identify the cause were unsuccessful.
One of the four prototype evaluations was incorrectly set up, resulting in participants performing the first task twice instead of the second task as intended. The issue was resolved on May 14th. As a result, any results before May 14th from this prototype were deemed invalid, and new participants had to be invited during the convenience sampling round discussed previously.

6.9.1 Design Variants

This section outlines the main differences between the design variants used in the evaluation.

	Edge	Vivaldi	A	B
Open new window	X			
Replace current window		X	X	X

Table 6.1: Behaviour when left-clicking on a list item. Edge opens a new window, while Vivaldi, A, and B replace the current window.

	Edge	Vivaldi	A	B
Open new window	X		X	
Replace current window		X		X

Table 6.2: Behaviour when left-clicking on a list item from an unsaved window. Edge and A open a new window, while Vivaldi and B replace the current window.

	Edge	Vivaldi	A	B
Inside three-dot menu	X			
Inside drop-down menu		X		
Next to unsaved window			X	X

Table 6.3: Process for saving a window. Edge hides the option behind a three-dot menu, Vivaldi uses a drop-down menu combined with a button, and the exploratory mockups A and B place it next to the unsaved window list item.

	Edge	Vivaldi	A	B
Interface closes	X	X		
Interface stays open			X	X

Table 6.4: Interface reaction after saving window. Edge and Vivaldi close the interface, while A and B keep the interface open.

	Edge	Vivaldi	A	B
Unsaved window is not displayed	X			
Unsaved window is displayed		X	X	X

Table 6.5: Displays current unsaved window in the interface. Vivaldi, A and B display the unsaved window at the top of the interface, while Edge does not.

	Edge	Vivaldi	A	B
Can change icon colour	X		X	X
Can change icon		X	X	X

Table 6.6: Customisation options. In Edge, users can change the colour of the icon. In Vivaldi, users can change the icon itself. In A and B, the user can change both.

	Edge	Vivaldi	A	B
"Stack" icon (blue)	X			
Randomised from set of icons		X		
Favicon of last visited site			X	X

Table 6.7: Default window icon. Edge uses a permanent "stack" icon, Vivaldi randomises from a set of choices, and browsers A and B use the favicon of the last visited site.

	Edge	Vivaldi	A	B
Workspace #	X			
New Workspace #		X		
Title of last visited site			X	X

Table 6.8: Default window names. Edge uses "Workspace #", Vivaldi uses "New Workspace #", and browsers A and B use the title of the last visited site. # is replaced by the number of workspaces the user has created.

	Edge	Vivaldi	A	B
Click triple-dot menu	X		X	X
Right-click list item		X	X	X

Table 6.9: Window context menu access process. Edge uses a triple-dot menu, Vivaldi uses a right-click on the list item, while browsers A and B support both methods.

Edge Mockup All references to "Invite to workspace" were removed, as well as the feedback buttons at the bottom of the interface. Removed additional menu options in context menus, such as "Learn more" and "Send feedback". The window background colour was accurately included, unlike in previous user tests where it was not. The visual indication for a selected window was also made to match the original Edge more closely. The onboarding screen was changed to be the same in all design variants, as the onboarding was not intended to be evaluated. In Edge's case, the onboarding was actually in opposition to the first user task, as it suggests that the user creates a new workspace when the task is to save their current window as a workspace.

Vivaldi Mockup Removed additional menu options in context menus, specifically "Copy All Links" and "Hibernate Workspace". The visual indication for a selected

window was made to match the original Vivaldi more closely. The onboarding screen was changed to be consistent across all mockups, as the onboarding was not intended to be evaluated.

6.9.2 Evaluation Protocol

The evaluation protocol involved testing four mockups, two exploratory prototypes, and two based on the Edge and Vivaldi browsers. Each participant was informed of and provided consent for the study's data collection in accordance with data protection regulations. Participants also had the option to volunteer a recording of their test session, which allowed for a more flexible data collection while respecting participant comfort levels regarding privacy. Participants were randomly assigned to test one prototype in which they completed tasks and answered questions. Tasks were designed to be short and straightforward, allowing for easy measurement of performance metrics such as completion time, error rates, and user satisfaction. Users also answered questions about their experience in between tasks and answered a modified SUS at the end. Participants were restricted to conducting tests using their desktop computers, as mobile devices were considered too different from the intended user experience.

6.9.3 Evaluation Tasks And Questions

This section describes the evaluation tasks and questions used to assess the usability of the different design variants when performing tasks using the tool. Additionally, at the end of each section, there was an optional feedback text block, including one at the end of the completed evaluation.

1. Saving a window on the first attempt
    - Task: Try saving the window using the icon in the top left.
    - Scale: Finding how to save the window was easy for me.
    - Scale: I feel confident that my window was saved.
2. Switching between an unsaved and a previously saved window
    - Task: Try switching back to the window you saved before.
    - Scale: I feel like some of my tabs may have been lost.
    - Scale: I liked that my videos opened in a new window - instead of the same window.
3. Saving a window on the second attempt
    - Task: Try saving your vacation window.
    - Scale: Finding how to save the window was easy for me.
    - Scale: I feel confident that my window was saved.
4. Clarity of default Icons and Names
    - Scale: I think these names make it easy to understand which window is related to "videos" and which is related to "vacation".
    - Scale: I think these icons make it easy to understand which window is related to "videos" and which is related to "vacation".
    - Multiple Choice: To make it easier for myself, I would change...
        - the icons

    - the names
    - nothing, I am satisfied with the current names and icons
5. Switching between two previously saved windows
    - Task: Try switching back to the video window you saved before.
    - Scale: I liked that my videos opened in the current window - instead of in a new window.
6. Opening a saved window in a new window, on prototypes where this is not the default action
    - Task: Try making it so that you have two separate windows open at the same time.
    - Scale: Finding how to open both windows at the same time was easy for me.
7. Overall impression of the experience
    - Scale: I think that I would like to use this system frequently.
    - Scale: I found the system needlessly complicated.
    - Scale: I thought the system was easy to use.
    - Scale: I think that I would need the support of a technical person to be able to use this system.
    - Scale: I found the various functions in this system to work well together.
    - Scale: I thought there was too much inconsistency in this system.
    - Scale: I would imagine that most people would learn to use this system very quickly.
    - Scale: I found the system very inefficient to use.
    - Scale: I felt very confident using the system.
    - Scale: I needed to learn a lot of things before I could get going with this system.

6.9.4 Things That Were Not Measured or Evaluated

- Process of renaming windows
- Process of editing icons
- Process of creating new saved windows
- Other designs for saving windows, such as dividing Vivaldi's new workspace button into two to improve clarity
- Placement of recently saved window in the list, top or bottom of the list
- Effect of showing name, colour, and icon on windows
- Effects of and content of context menu on right-clicking a tab
- Dragability of list-items and how it is influenced by drag-handles
- And many more potential variations of the design

7
Competitor Analysis

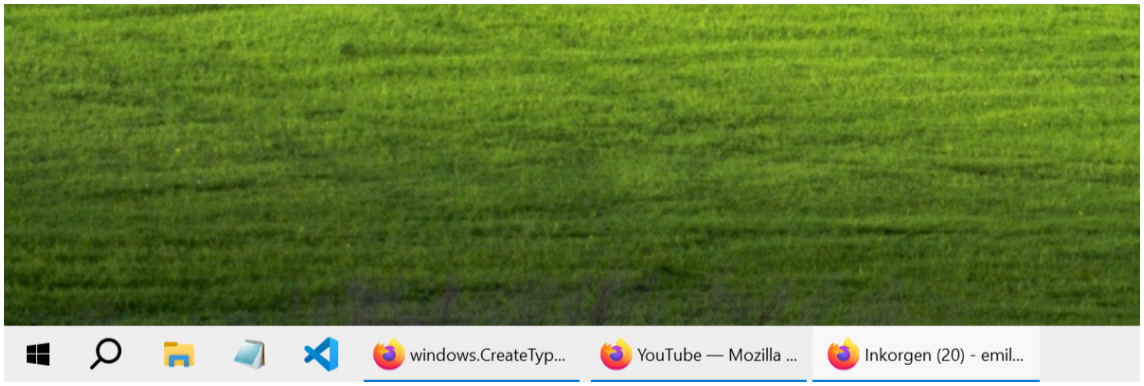
A competitor analysis was used to examine how major web browsers support window restoration and tab organisation. An initial set of approximately ten popular browsers was reviewed. From these, Microsoft Edge and Vivaldi were selected for deeper analysis based on two main criteria: the presence of native window-saving features and compatibility with Windows 10. In addition, the most widely used Firefox extension offering workspace functionality, Simple Tab Groups, was included to represent Firefox's current capabilities when improved by extensions. The chapter begins with an overview of Firefox in its original state, without extensions.

7.1 Firefox

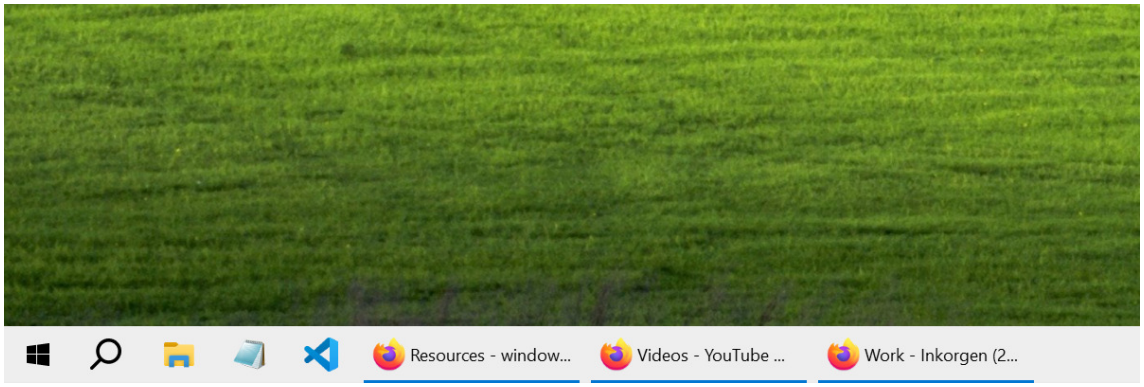
Firefox meets basic modern browser standards by automatically restoring all open windows if the program was closed and reopened, preserving their previous state. However, this method requires that the user does not close any windows and instead keeps them open indefinitely. Firefox does not offer a dedicated "Save Window" function, which could allow them to close windows and reopen them later while also giving them peace of mind that their tabs are safe. Instead, users have to rely on workarounds; while features like bookmarks, search history, and manual window restoration can be used to help recover closed windows, they are not designed specifically for this purpose and are cumbersome to use. The task analysis below shows that these workarounds do not offer a practical solution for reliably and intentionally saving windows; see 8.

Extensions for Firefox Firefox's functionality is lacking in comparison to its competitors, but it can be enhanced with add-ons that improve tab management:

- Window Renaming: Some extensions let users manually rename windows, which helps in organising and identifying them [37], [38].
- Side-by-Side Tab Viewing: Certain extensions enable a tab to be viewed in a sidebar alongside the main window [39], [40], [41]. However, the fixed sidebar size limits the effectiveness of these extensions.
- Tab Management Limitations: No extension can fully support tab stacking or grouping, as extension developers are restricted from modifying the visuals of the tab bar beyond the tab titles and favicons.

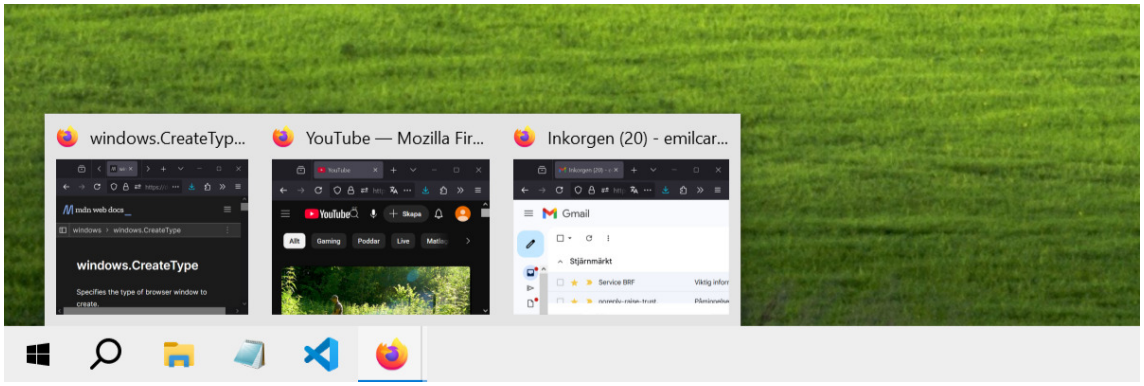


(a) Unnamed windows in the taskbar.

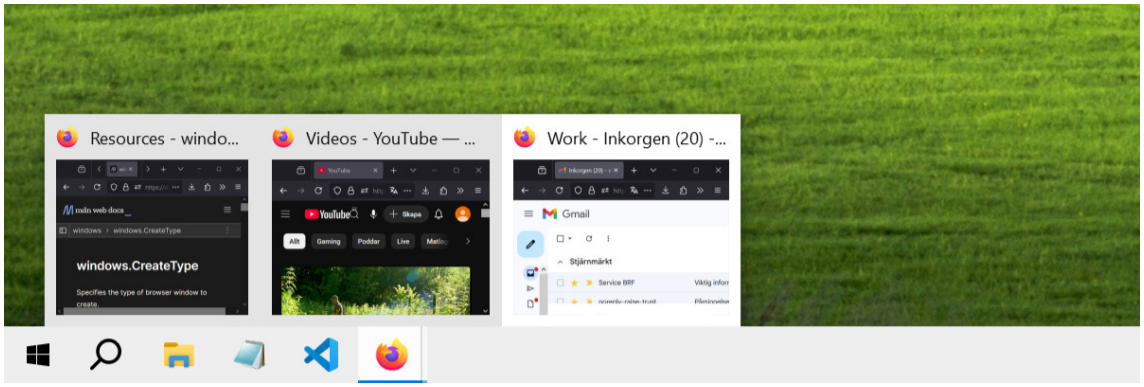


(b) Named windows in the taskbar.

Figure 7.1: Comparison between named and unnamed windows in the taskbar. User operating system setting for the taskbar is to show labels.



(a) Unnamed windows in the taskbar.



(b) Named windows in the taskbar.

Figure 7.2: Comparison between named and unnamed windows in the taskbar. User operating system setting for the taskbar is to hide labels.

7.2 STG (Simple Tab Groups)

STG is a popular Firefox add-on that replicates workspace functionality by allowing users to save, name, and switch between windows [42]. Once a window is saved, any changes to its tabs are automatically recorded, and its assigned name and icon appear in the extension popup for easy identification. The name of a saved window can be used to identify it in the taskbar; see the difference between named and unnamed windows in the taskbar in figure 7.1. This functionality is not unique to STG, as it works the same way when using Edge, Vivaldi, or any other extension to rename windows. However, the user can change a setting at the operating system level to hide the labels in the taskbar, which requires them to hover over the taskbar icon to see the names of the windows, see figure 7.2. Note that there are other ways to view open windows in Windows 10, such as using the keyboard shortcut Alt+Tab to display a grid of open windows where each window is also labelled with its name.
The extension displays a popup list showing each saved window along with its name, icon, and colour. Clicking an item replaces the current window's tabs with those from the selected window. Right-clicking provides an option to open the saved window in a new window. If the saved window is already open, it is brought to the foreground.
From the extension popup, the user can also create a new group. If the current window is already saved, the button will create a new empty group. If the current window is not saved, the new group will contain all of the tabs of the current window. The user can also search for open tabs using a search bar.

Settings STG offers a lot of settings targeted towards experts. The settings allow users to control how windows are saved and resumed, including synchronisation with bookmarks and adjustments to popup behaviour. Users can customise context menus for both individual tabs and entire groups, fine-tune how groups are created and managed, and modify the tab overview display. In addition, the extension provides many keyboard shortcuts to streamline navigation and control, providing expert users with extensive options for customising their workflow.

7.3 Edge

Edge lets users save entire windows as workspaces. A saved workspace retains all its tabs, making it easy to switch between multiple projects.
When you save a workspace, its name shows up in three places: in the workspace popup, next to the edge workspaces icon button that opens the popup, and on the desktop taskbar. The user can also assign a colour to a workspace, which changes both the icon in the list of workspaces and the colour of the window itself.

Description of Workspaces Functionality Edge's Workspaces appear in a popup, where each workspace (saved window) can be reopened with one click. Each workspace is named by the user upon creation and listed with its name and tab count; see figure 6.1. The selected workspace appears brighter and has a blue line

on the left. Users can drag to reorder workspaces, although there is no visual cue for this. An icon on the left of each workspace has no practical use apart from displaying the colour previously selected by the user. Saving windows requires the user to click ••• horizontal three-dot icon and navigate a short list of options (right-clicking a workspace does not open this context menu), where they will find the option to save the window. Creating a new workspace, which does not contain any tabs, does not require the user to navigate a menu; instead, it is presented directly as a + add icon button.

When a workspace is selected, three buttons appear at the top of the popup: Invite to Workspace, Edit Workspace, and Delete Workspace. These options are also available by clicking the ••• horizontal three-dot icon on the right. The Invite to Workspace option lets users add collaborators, the Edit Workspace option opens a modal to rename and change the workspace colour, and the Delete Workspace option removes the workspace.

When a workspace is clicked, its action depends on its current state: if it is already open, the window is brought to the front; if not, a new window is opened to display the workspace.
The popup also includes a + add icon button for creating new workspaces. When clicked, users are prompted to name the workspace and, optionally, select a colour. If the user does not name the workspace, it will automatically be called "Workspace #", where # increases for every unnamed workspace. The new workspace then appears in the list and opens automatically in a new window.
If the user has not saved their current window, the ••• horizontal three-dot icon button next to the + add icon button will have an additional option to move all tabs to a new workspace. At this point, the user names and optionally colours the new workspace. When they press Done, the current window is promoted to a workspace while retaining all its tabs.

Additional Features Edge includes several additional features that improve tab management. While these features are not directly related to workspaces, they can help users manage their tabs more effectively, which reduces cognitive load and potentially the number of windows they need to keep open. These features include:

- Side-by-Side Tab Viewing: Users can split the screen to display two tabs simultaneously and open a tab in a sidebar that remains visible as they browse the web. This reduces the need for multiple windows to view content side by side.
- Tab Grouping: Tabs can be grouped in the tab bar; these groups can be opened or closed to save space in the tab bar. This keeps the tab bar uncluttered and organised to reduce cognitive load.

Settings Edge offers limited settings for Workspaces. Users can either disable the feature entirely or choose which workspace should open external links. Disabling Workspaces hides the popup and removes the edge workspaces icon button from the tab bar. No other settings are available.

7.4 Vivaldi

Vivaldi also offers a built-in workspace feature for saving windows. Additionally, it includes tab grouping and side-by-side viewing, which works similarly to Edge but with more customizability for expert users.

Description of Workspaces Functionality Vivaldi displays workspaces in a popup list with name, icon and the number of open tabs; see figure 6.2. Each saved workspace has a `= drag handle icon` to indicate that they can be reorganised, except for the active workspace, which displays a `✓ checkmark icon`. The saved workspace has an icon on its left side, which can be customised by clicking it. Right-clicking a workspace opens a context menu that allows the user to delete, rename, or open the workspace in a new window.
Clicking a workspace behaves differently based on its state. If the workspace is already open in another window, that window is brought to the foreground. Otherwise, the current window updates to display the workspace's tabs, replacing any open tabs. Unsaved tabs remain accessible as an extra list item at the top of the popup, separated by a horizontal line.
This extra item shows unsaved tabs from the current window. When unsaved tabs exist, it is named after the last open tab. If there are no unsaved tabs, the item remains unnamed and shows a count of zero tabs, which happens only after opening a saved workspace in a new window.
Clicking the extra list item allows the user to switch between the window's unsaved tabs and any saved workspaces. However, the unsaved tabs are specific to that window and will not appear in other windows. They will be lost if the window is closed.

Vivaldi allows users to create new workspaces via the `New Workspace` button at the bottom of the popup. When clicked, the user is asked to name the new workspace. If the user does not name the window, it will be called "New Workspace #". Once created, the workspace appears in the list with zero tabs.
If the user has unsaved tabs open, the `New Workspace` button is sided by an `⌄ dropdown icon`. Clicking the `⌄ dropdown icon` opens a menu where the user can choose to create a new empty workspace or one that includes the unsaved tabs.

Additional Features Vivaldi includes several additional features that improve tab management. While these features are not directly related to workspaces, they can help users manage their tabs more effectively, which reduces cognitive load and potentially the number of windows they need to keep open. These features include:

- Side-by-Side Tab Viewing: Users can view multiple tabs next to each other, with support for more than two tabs at a time. They can also open tabs in a sidebar that remains on the screen while browsing. These features reduce the need for multiple windows to view content side by side.
- Tab Grouping: Vivaldi offers three modes for organising tabs into groups in order to reduce clutter and improve organisation:
  - Two-level Navigation: Grouped tabs appear on a secondary line below the main tab bar.

- Accordion System: Grouped tabs expand and collapse, similar to the Edge browser.
- Compact Mode: Clicking on a group opens a menu that allows the user to select which tab to navigate to.

Settings Vivaldi does not offer a settings menu directly in the workspaces popup. Instead, workspace settings are available in the browser's general settings. Users can disable the feature entirely to hide it from the tab bar and configure keyboard shortcuts for switching between workspaces, creating new ones, and selecting a specific workspace, among other options.

7.5 Analysis of Competitor Analysis

The competitor analysis shows that Firefox lacks features for saving and resuming tasks compared to Edge and Vivaldi. While both Edge and Vivaldi have implemented systems for saving and resuming tasks, Firefox users are left with the option of keeping windows open, bookmarking all open tabs, or restoring recently closed windows.

Native Workspace Systems in Edge and Vivaldi Edge and Vivaldi both offer a workspace system for saving windows with open tabs, allowing users to easily resume their work. This is a significant improvement over Firefox's current methods, which do not provide a dedicated solution for saving windows.
Edge opens each workspace in a new window, helping to keep tasks separate by providing a clear mental model; however, this approach can lead to many open windows. Vivaldi, on the other hand, replaces the current window with the new workspace, which can be more efficient for users who prefer this setup. While Edge's method is ideal for users who need a distinct separation between projects, it requires extra effort to manage and navigate multiple open windows effectively.

Third-Party Solutions in Firefox Some Firefox extensions, such as Simple Tab Groups (STG), offer similar functionality to Edge and Vivaldi, but they are not built into the browser itself. This means that users must rely on third-party extensions to achieve the same level of functionality as Edge and Vivaldi. This is not ideal, as users may not be aware of these extensions or may not want to rely on them for essential functionality. Additionally, these extensions may not be as well designed or integrated into the browser as built-in features, leading to a less seamless user experience.

Additional Tab Management Features The analysis also shows that Edge and Vivaldi have implemented additional features for tab management that are not available in Firefox. These features include the ability to group tabs in the tab bar and the ability to view tabs side by side. These features can help users manage their tabs more effectively, reducing both cognitive load and, in some cases, the number of windows they need to keep open (see 9.1).

While Firefox extensions offer some of these features, such as basic side-by-side tab viewing, they are not as well-integrated into the browser as the built-in features of Edge and Vivaldi. One reason for this is that Firefox imposes limitations on the functionality of extensions, which can make it challenging for developers to create features that are as seamless and well-integrated. While there is good reason not to allow extensions to have full access to the browser, this limitation does restrict the potential of extensions to provide the same level of functionality as built-in features.

Customization Options for Workspaces Furthermore, allowing users to choose both icons and colours for their workspaces offers a practical benefit. While Edge restricts customisation to colours and Vivaldi limits it to icons, STG provides both options. This extra level of flexibility could help users identify their workspaces more easily at a glance by tailoring visual cues to their personal preferences. Users can choose a colour and icon that represents the task to them, helping them quickly identify the right workspace when multiple ones are open. Firefox (without extensions) offers no ability to name or customise the appearance of windows.

Summary of Competitor Analysis The competitor analysis shows that Firefox is missing several built-in features that help save and resume tasks, especially when compared to Edge and Vivaldi.
Both Edge and Vivaldi have their own workspace systems designed to save sets of open tabs. Edge opens each workspace in a new window, which keeps different tasks separate; however, this can result in many open windows. On the other hand, Vivaldi updates the existing window, which might be a more streamlined approach for some users.
Additionally, Edge and Vivaldi have implemented other features for tab management that are not present in Firefox. These can help users manage their tabs more effectively and reduce cognitive load, reducing the need for multiple windows.
While Firefox does have some extensions that offer similar functionality, such as STG, they are not as well-integrated into the browser as the built-in features. This means that users may not be aware of these extensions or may not want to rely on them for essential functionality.
In summary, the absence of a native tool in Firefox for saving or customising windows forces users to rely on workarounds or extensions, which often function poorly.

8

Task Analysis

This chapter presents a task analysis comparing how Firefox (with and without extensions), Microsoft Edge, Vivaldi, and two design prototypes support window and session management. Five representative tasks were selected to reflect how users save and resume work in a web browser, such as saving a window, switching between sessions, and restoring saved windows. Each system was evaluated by simulating the steps needed to complete these tasks. The same five tasks were applied to STG, Edge, Vivaldi, and the two prototypes for consistent comparison. Because Firefox does not support workspaces natively, equivalent workarounds were tested instead.

8.1 Firefox

This section examines how users can save tabs in Firefox without relying on extensions. Three features in Firefox are sufficiently similar to classical saving functionality to be considered for analysis: saving all open tabs as bookmarks, keeping windows open indefinitely, and restoring windows after closing them. Other workarounds for saving tabs will not be discussed in this analysis.

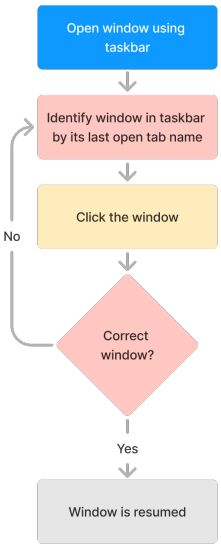


Figure 8.1: Process of resuming a window using the taskbar.

Leaving a Window Open to Avoid Losing Tabs This process can be seen in figure 8.1. A simple way to maintain work progress is to keep the window open. Since modern browsers automatically restore open windows, this method is common among users. Keeping the window open preserves its state, removing the need for recovery actions.

To resume work in an open window, the user first identifies it on the taskbar by its last open tab name. With multiple windows or tabs, this can be challenging as each additional window increases the cognitive load. After clicking on a window, the user checks if it is the expected one. If not, they repeat the identification process until the correct window is found, and work can resume.

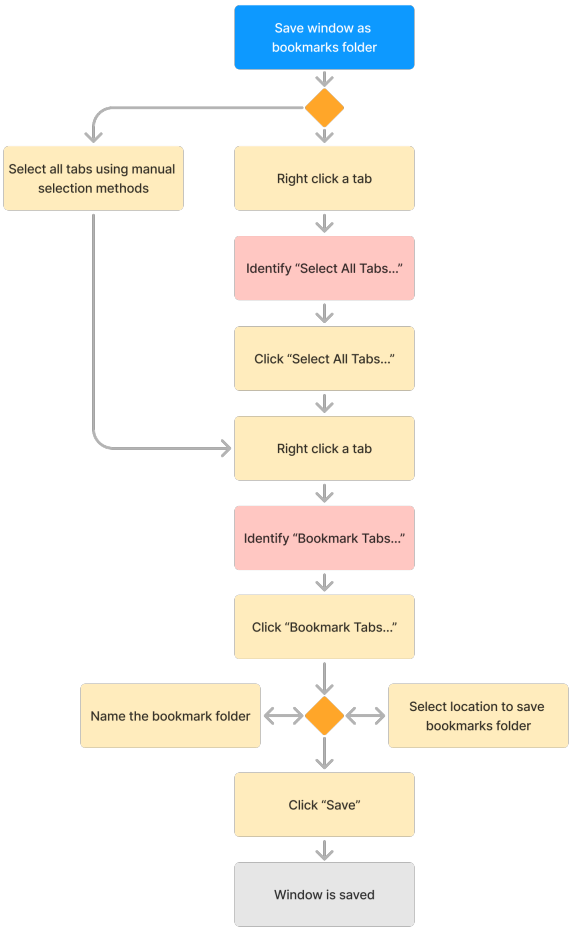


Figure 8.2: Process of saving all tabs in a window as a folder of bookmarks.

Saving a Window using Bookmarks This process can be seen in figure 8.2. This approach involves two steps: first, bookmarking all open tabs into a folder, and second, reopening that folder later to resume the session. While this method safely stores tabs for long-term use, it is not automated; users must manually and repeatedly save their session state to avoid losing tabs.
It is important to note that the user cannot overwrite previous saves. This means the user must manually manage their saved windows, for example, by deleting old saves or moving them to an archive folder. Other approaches, such as continuously saving tabs as bookmarks, can achieve a similar result.
The main issue with this approach is that it does not work as an auto-save feature. If a new tab is opened after saving the window as a folder of bookmarks, the new tab will not be included in the saved state. Although some users might prefer this method to save only essential tabs while ignoring temporary ones, this workflow is already supported, so it will not be considered further.

To use this method, the user first saves all open tabs as a bookmark folder. There are two ways to do this. The first is a six-step process that requires navigating menus in multiple stages, though some steps can alternatively be performed by manually selecting tabs and using a context menu. The second option is using a shortcut to achieve the same result in fewer steps. Although neither option is clearly presented to the user, the second option is not displayed anywhere and can be most easily found by searching for help online.

Next, a popup appears with options for saving the group of tabs as a folder. Here, the user can optionally name the folder and choose or create a parent folder for the bookmarks. The naming step requires typing a name, and selecting a location typically involves multiple steps. The process is completed when the user clicks the `Save` button.

The simplest process takes two steps: using the keyboard shortcut and clicking `Save`. However, this approach results in an unorganised bookmarks folder, making it hard to find specific sessions. A more practical process includes the third step of naming the folder. Users must also spend time removing or archiving old save folders, adding further steps to maintain organisation.

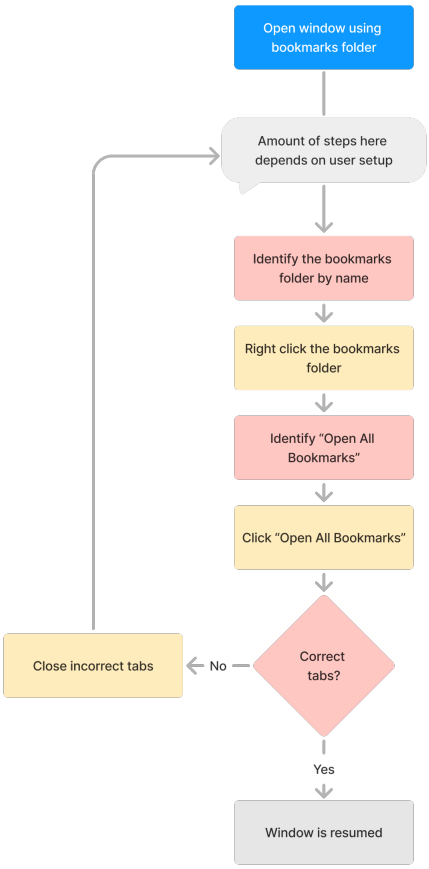


Figure 8.3: Process of resuming a window which was saved as a folder of bookmarks.

Restoring a Window using Bookmarks This process can be seen in figure 8.3. To resume a window after saving its tabs and closing it, the user first navigates to the folder containing the saved bookmarks. This folder might be immediately visible in the bookmarks toolbar or may require opening the bookmarks sidebar using `ctrl + b` and locating the correct folder. Next, a four-step process begins, which should result in opening the correct tabs.

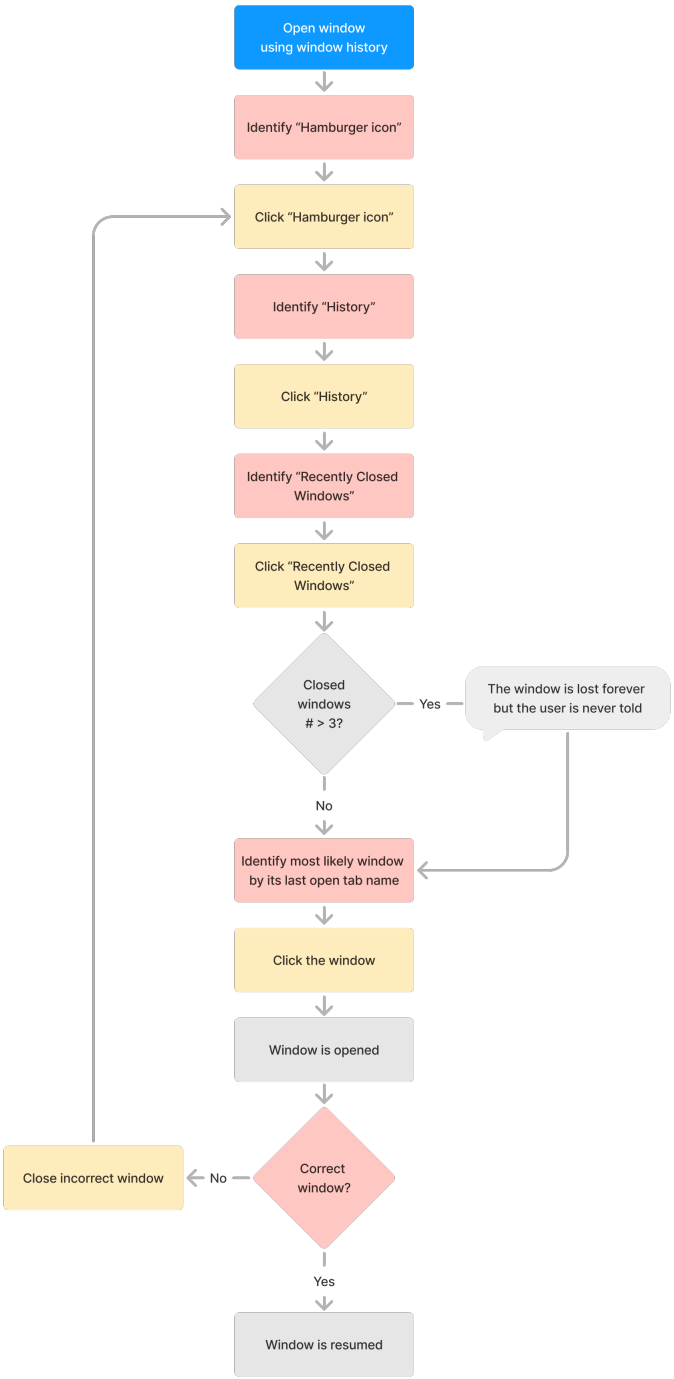


Figure 8.4: Process of resuming a window using the window history.

Restoring a Window using Recently Closed Windows This process can be seen in figure 8.4. When a window is closed, whether intentionally or accidentally,

users can restore it through the browser's history feature.
The recovery process begins with the user clicking the ≡ three-line icon in the browser to open a context menu. The user must then locate and select the History option to access the browser's history. Here, the user can find Recently Closed Windows. Clicking this displays a list of closed windows, each named after the tab that was active when the window closed.
By default, only three windows can be stored using this method. Although additional windows can be saved by adjusting hard-to-access settings, if more than the maximum amount of windows are closed, any additional windows are permanently lost.
After identifying the desired window, the user clicks it to resume the session. Once reopened, the user checks if it is the correct window. If not, the user closes the incorrect window and repeats the process. The goal is achieved when the correct window is restored, allowing the user to resume work.

8.2 Firefox with STG

This section evaluates Simple Tab Groups (STG), the primary Firefox extension that adds workspace-style tab saving and restoration.

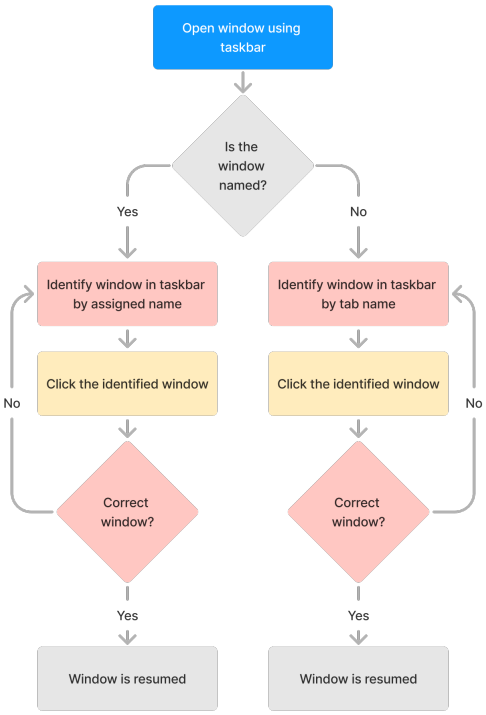


Figure 8.5: Process of resuming a window using the taskbar when using workspaces.

Resuming a Window using Taskbar This process can be seen in figure 8.5. STG (Simple Tab Groups) follows a similar approach, where saving a window as a workspace updates its taskbar title to the workspace name, which aligns with the functionality observed in Edge and Vivaldi.

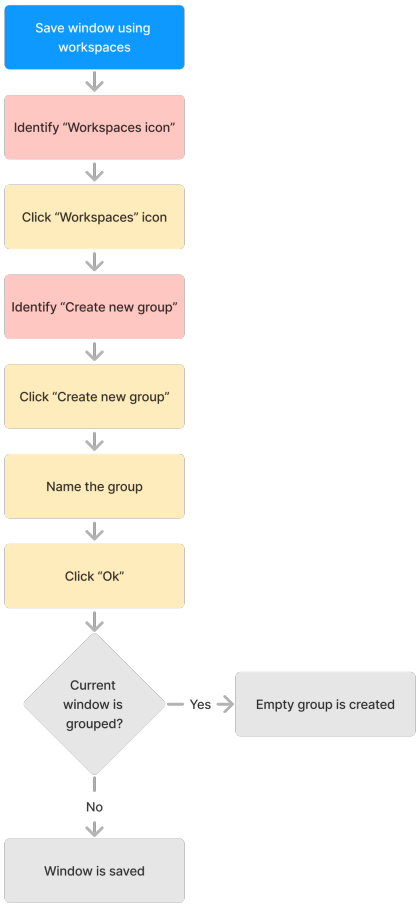


Figure 8.6: Process of saving a window using workspaces in STG (Simple Tab Groups).

Saving a Window using Workspaces This process can be seen in figure 8.6. STG implements a more streamlined process for saving windows compared to Edge and Vivaldi. After clicking the stg workspaces icon to open the popup displaying all saved workspaces, the user can click the button. If the current window is already saved, clicking the button will create a new empty workspace; if not, it will save the current window as a workspace. The user then optionally names the workspace, and after clicking Ok, the workspace is created. Customisation options, including colour and icon changes, can be made after creation.

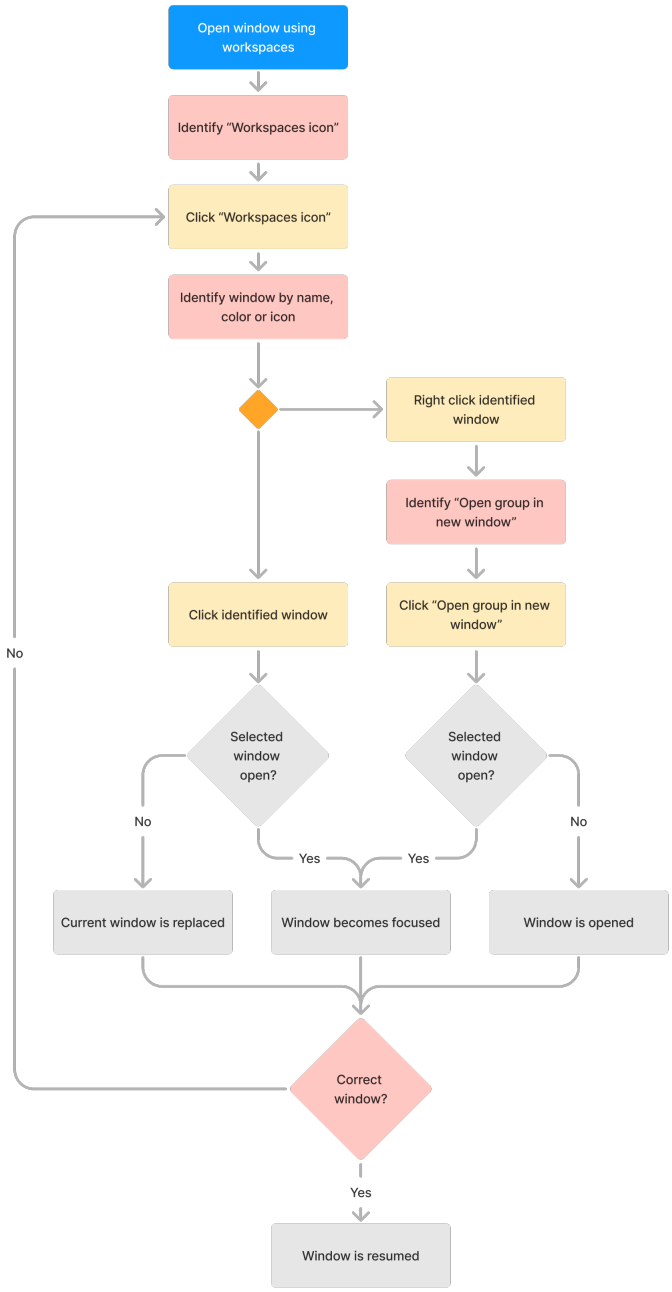


Figure 8.7: Process of resuming a window using workspaces in STG (Simple Tab Groups).

Restoring a Window using Workspaces This process can be seen in figure 8.7. Restoration in STG works the same as in Vivaldi. When the user clicks a workspace to open it, if that workspace is already open in another window, that window is brought to the foreground. If the workspace is not already open in another window,

it will replace the current window. Alternatively, users can right-click the workspace to open it in a new window.

8.3 Edge

Edge includes a native Workspaces feature; this section examines how it manages saving and restoring windows.

Resuming a Window using Taskbar This process can be seen in figure 8.1. Leaving a window open when using workspaces works similarly to Firefox. In Edge, when a window is saved as a workspace, its title in the taskbar changes from the current tab name to the workspace name. This change helps users quickly identify the correct window from the taskbar without needing to remember the last open tab; see figure 8.1 compared to figure 8.5. This functionality is consistent with other browsers that support workspaces.

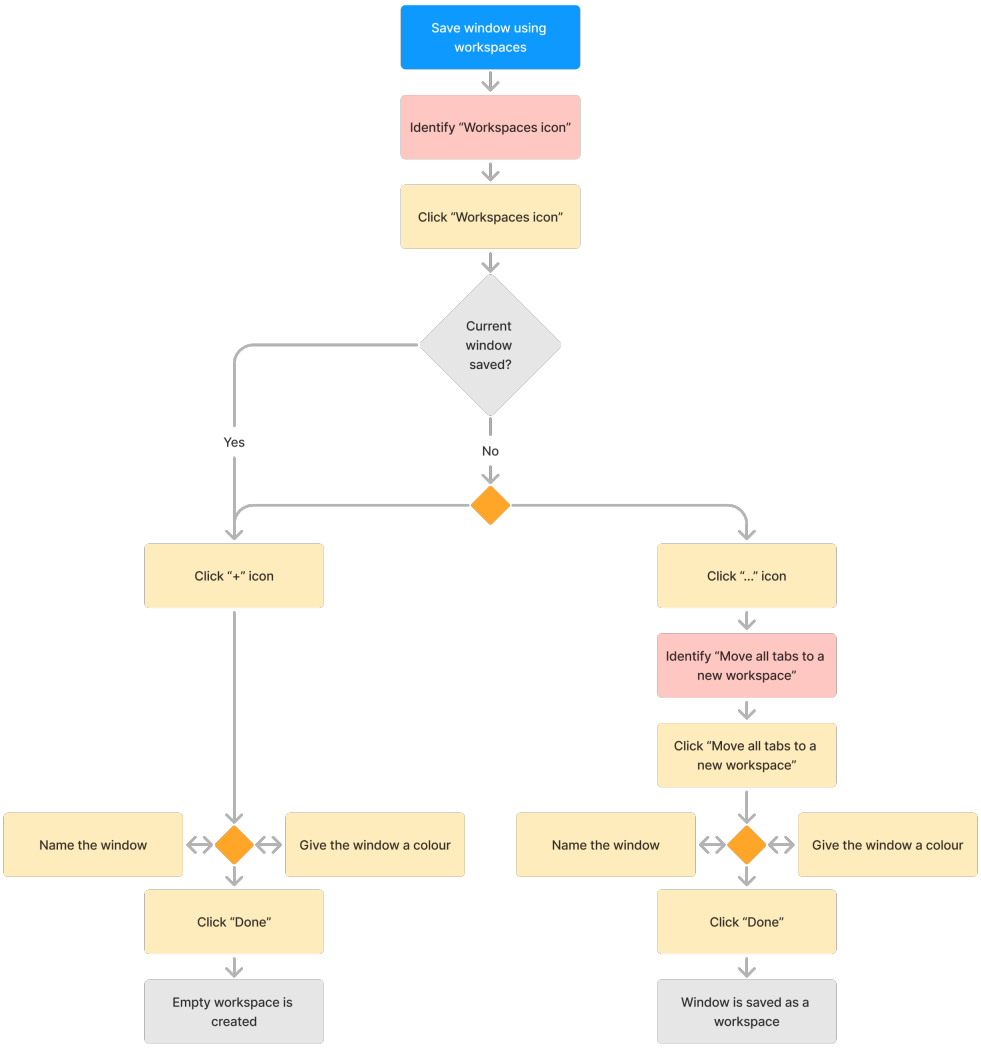


Figure 8.8: Process of saving a window using workspaces in Edge.

Saving a Window using Workspaces This process can be seen in figure 8.8. The process begins when the user clicks the edge workspaces icon to open the workspaces popup. In this popup, users can either save the current window as a workspace or create a new empty workspace. To create a new workspace, the user clicks the + add icon button, which prompts them to name the workspace and choose a colour. If a name is not chosen, a default name will be given. Alternatively, to save the current window and its tabs, the user must click the discrete ••• horizontal three-dot icon next to the + add icon button and select the option Move all tabs to a new workspace.

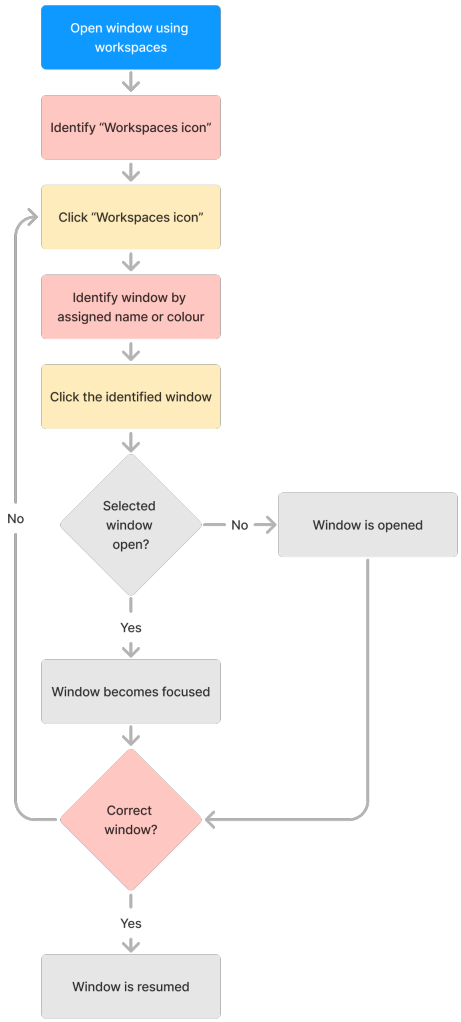


Figure 8.9: Process of resuming a window using workspaces in Edge.

Restoring a Window using Workspaces This process can be seen in figure 8.9. The user clicks the edge workspaces icon to open a popup listing saved workspaces and then selects one. If the workspace is already open in a window, that window is brought to the foreground; if not, the workspace opens in a new window.

8.4 Vivaldi

Vivaldi includes a native Workspaces feature; this section examines how it manages saving and restoring windows.

Resuming a Window using Taskbar This process can be seen in figure 8.5. In Vivaldi, the task of resuming a window using the taskbar functions the same as in Edge. When a window is saved as a workspace, the title shown in the taskbar changes to the workspace name. This design choice enables users to quickly identify the desired workspace without needing to recall the name of the last active tab.

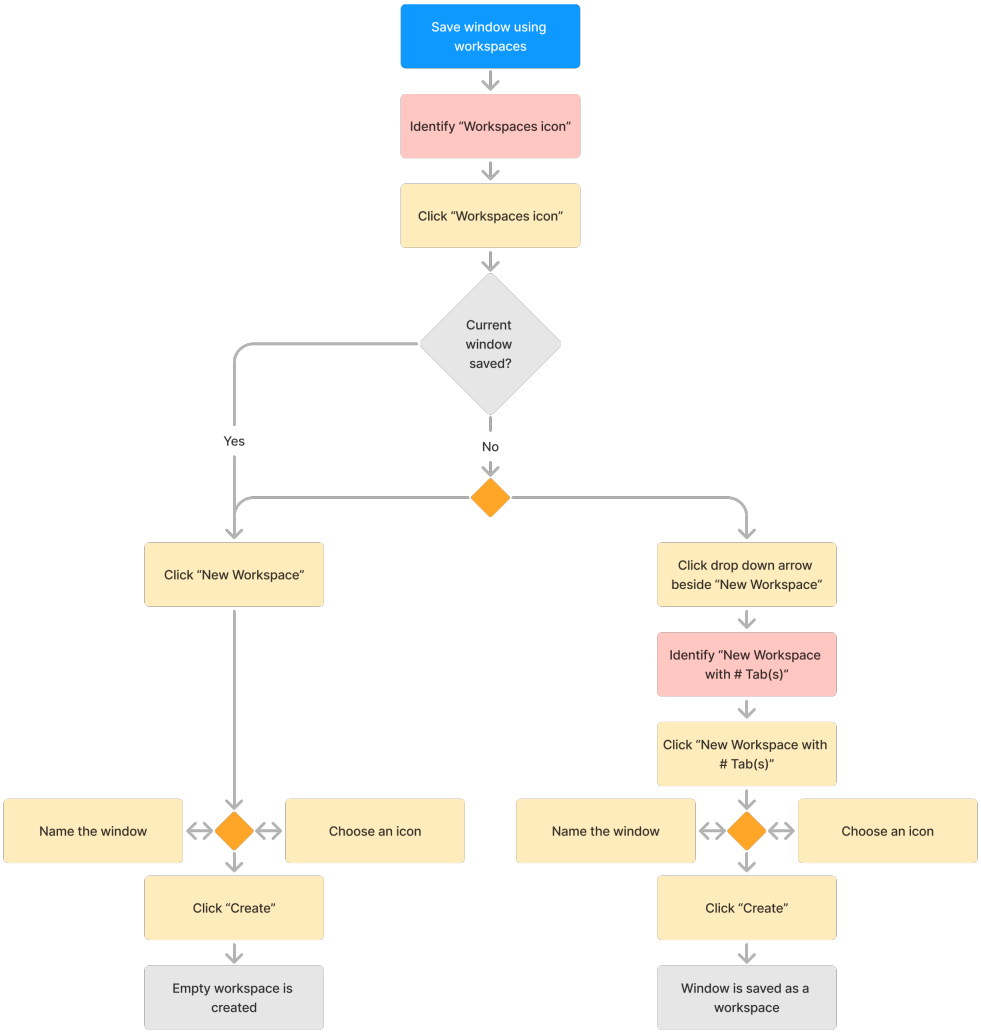


Figure 8.10: Process of saving a window using workspaces in Vivaldi.

Saving a Window using Workspaces This process can be seen in figure 8.10. Vivaldi's process for saving windows is similar to that of Edge. The process starts when the user clicks the vivaldi workspaces icon to open the workspace popup. In Vivaldi, the user can click the New Workspace button to create a new empty workspace, at which point they will be given the option to name their workspace and give it an icon or to directly click Create. To save the current window and

any open tabs, the dropdown icon adjacent to the New Workspace button can be clicked, at which point the user can select New Workspace with # Tab(s), at which point the user can optionally name and choose an icon for the workspace.

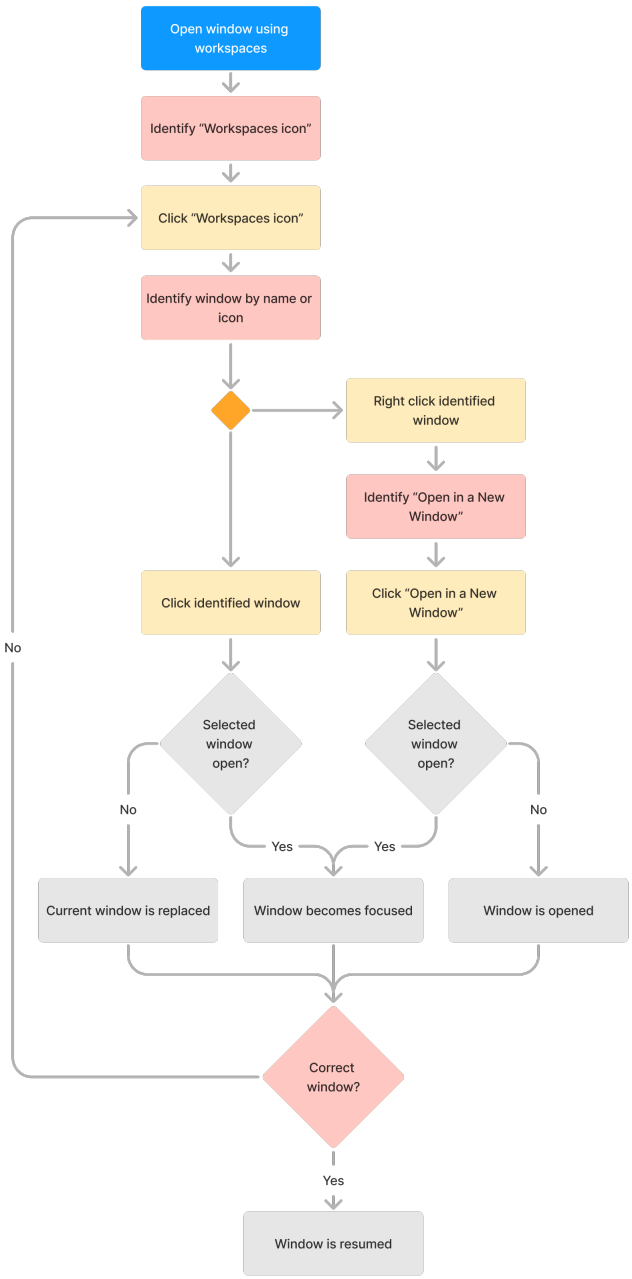


Figure 8.11: Process of resuming a window using workspaces in Vivaldi.

Restoring a Window using Workspaces This process can be seen in figure 8.11. The user clicks the workspaces icon to display a list of saved workspaces. When the user clicks a workspace to open it, if that workspace is already open in another window, that window is brought to the foreground. If the workspace is not already open in another window, it will replace the current window. Alternatively, right-clicking a workspace presents an option to open it in a new window.

8.5 Experimental Prototype A

Prototype A is one of the two prototypes developed for the evaluation. Prototype A differs from Prototype B by adopting part of Edge's window restoration functionality. When the user switches to another saved workspace, the user's current window is replaced if it was already saved; if it has unsaved changes, the new workspace opens in a separate window, leaving the unsaved window where it was. This design enables users to switch quickly between saved workspaces without creating additional windows and preventing users from feeling that they might have overwritten an unsaved window.

Resuming a Window using Taskbar This process can be seen in figure 8.5. Resuming a window using the taskbar works the same as in Edge, Vivaldi and Experimental Prototype A.

Saving a Window using Workspaces This process can be seen in figure 8.13. When the user clicks the experimental workspaces icon, a popup appears displaying all saved windows. If the current window is not already saved, a button appears next to it in the list of workspaces. Clicking this button instantly saves the window. To create a new empty workspace, the user clicks the + add icon. This action presents a choice between New saved window, which opens a new empty workspace in a new window and Save this window, which saves the current window and its tabs.
After the workspace is saved, the user has the option to change its name, icon, and colour.

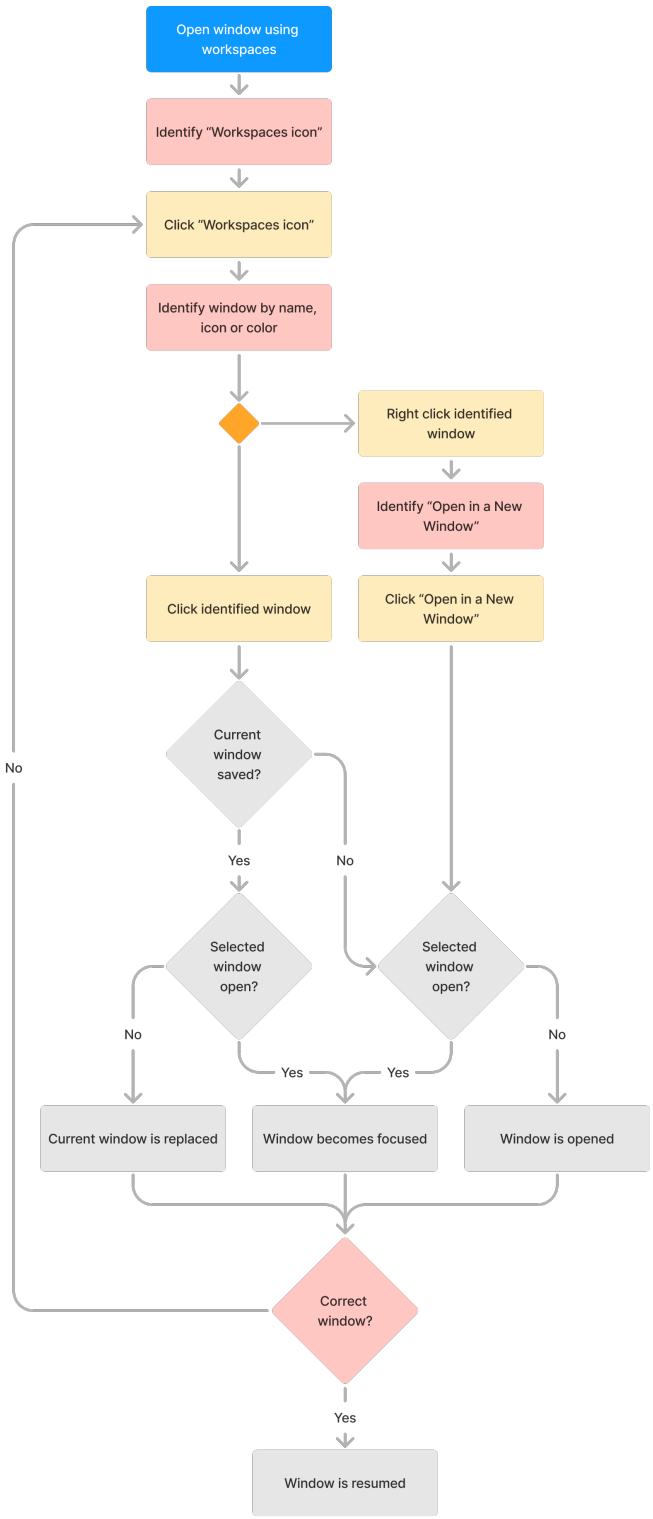


Figure 8.12: Process of resuming a window using workspaces in experimental prototype A.

Restoring a Window using Workspaces This process can be seen in figure 8.12. The process for restoring a window differs slightly from Prototype B, as it borrows some inspiration from Edge. When the user clicks a workspace to open it, if the workspace is already open in another window, that window is brought to the foreground. If it is not open, the selected workspace will be opened based on the state of the current window: if the current window is already saved, the selected workspace will replace it, but if the current window is not saved, the workspace will open in a new window. As with Prototype B, right-clicking the workspace offers the option to open it in a new window.

8.6 Experimental Prototype B

Prototype B is one of the two prototypes developed for the evaluation. Prototype B differs from Prototype A (see section 8.5) in that switching to another saved workspace always replaces your current window, whether it is saved or not. This keeps the user working in a single window but risks scaring the user as they think their window has been overwritten.

Resuming a Window using Taskbar This process can be seen in figure 8.5. Resuming a window using the taskbar works the same as in Edge and Vivaldi.

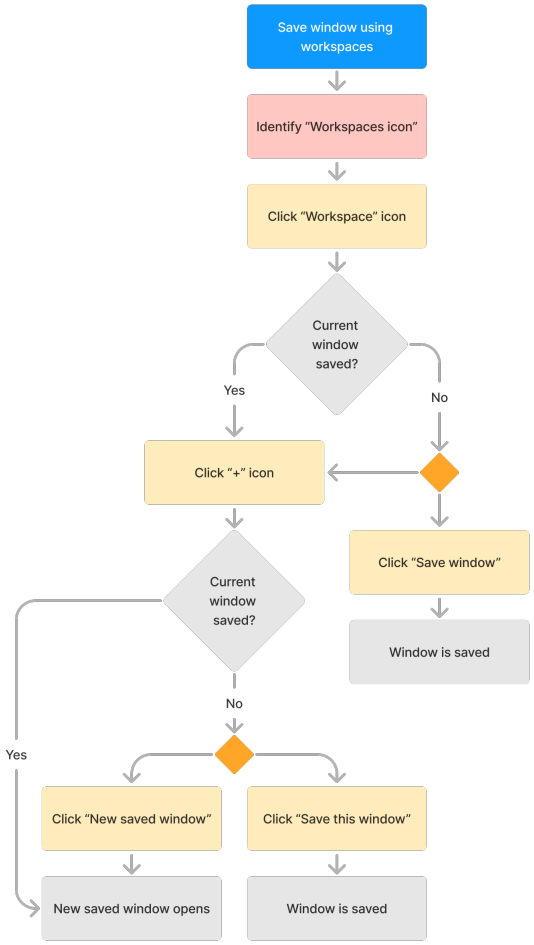


Figure 8.13: Process of saving a window using workspaces in the experimental prototypes.

Saving a Window using Workspaces This process can be seen in figure 8.13. When the user clicks the experimental workspaces icon, a popup appears displaying all saved windows. If the current window is not already saved, a button appears next to it in the list of workspaces. Clicking this button instantly saves the window. To create a new empty workspace, the user clicks the + add icon. This action presents a choice between New saved window, which opens a new empty workspace in a new window and Save this window, which saves the current window and its tabs.
After the workspace is saved, the user has the option to change its name, icon, and colour.

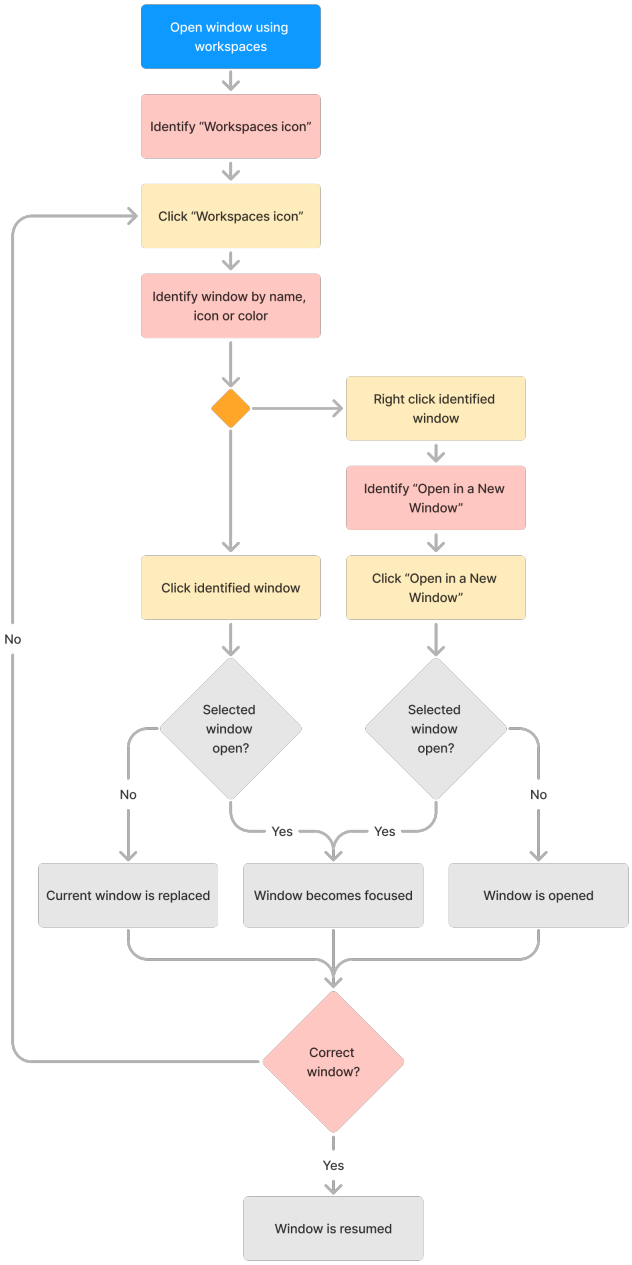


Figure 8.14: Process of resuming a window using workspaces in experimental prototype B.

Restoring a Window using Workspaces This process can be seen in figure 8.14. Restoring a window using workspaces works the same in Prototype B as in Vivaldi. When the user clicks a workspace to open it, if that workspace is already open in another window, that window is brought to the foreground. If the workspace

is not already open, it replaces the current window. Alternatively, right-clicking the workspace gives the option to open it in a new window.

8.7 Analysis of Task Analysis

In this section, the task flows from before are analysed by counting the mental effort and click count to show the relative strengths and weaknesses of each browser's and prototype's window-saving workflow. In the tables, mental work is marked in red, clicks are marked in yellow, and automatic steps are in grey.

Task	Resuming a window using the taskbar	Saving all tabs as a folder of bookmarks	Resuming a window from bookmarks	Resuming a window using the window history
Mental Work	2	2	3	4
Clicks	1	5	3	4

Table 8.1: Firefox: Comparison of mental work and click count for various tasks.

Firefox Specific Tasks Firefox's need for these workarounds is unique. Edge, Vivaldi, STG, and the prototypes eliminate these convoluted steps by offering a way to save windows.
The Firefox tasks in table 8.1 reveal that using the taskbar to resume a window requires minimal effort (2 mental work, 1 click - same as when using workspaces) but suffers from the inability to assign custom names for easier identification. Saving all tabs as a folder of bookmarks, although functionally available, demands more cognitive effort and extra clicks due to not being designed for the purpose, as well as the need for the user to create their own system of organisation. Resuming a window from such a bookmark folder increases both the thinking and clicking requirements, depending on the user's organisation. Finally, using the window history for resumption is the most inefficient option, requiring additional mental work and steps. Additionally, there is a limit of three windows being saved this way at a time, leading to a significant risk of the user accidentally losing an important window.

Prototype	STG	Edge	Vivaldi	B	A
Mental Work	2	2	2	2	2
Clicks	1	1	1	1	1

Table 8.2: Resuming a window using the taskbar: Comparison of mental work and click count for non-Firefox prototypes.

Taskbar Resumption When resuming a window using the taskbar, all prototypes in table 8.2 (STG, Edge, Vivaldi, B, A) require the same level of mental work

(2) and a single click to resume a window using the taskbar. The main advantage here is that workspaces allow users to assign window names, making the window easier to locate in the taskbar. A window that is easier to locate also means that the user is less likely to open the wrong window, which would require them to repeat the process of locating the correct window. This is a significant improvement over Firefox, where users must rely on the name of the last open tab to identify the correct window. This can be a problem when multiple windows are open, as the user may not remember the name of the last open tab. This is especially true if the user has multiple windows open with similar names.

Prototype	STG	Edge	Vivaldi	B	A
Mental Work	0	1	1	0	0
Clicks	3	4	4	2	2

Table 8.3: Saving a window using workspaces: Comparison of mental work and click count for non-Firefox prototypes.

Saving Windows with Workspaces When saving a window using workspaces, see table 8.3, the process differs slightly between prototypes. STG, B, and A are efficient, requiring little to no mental effort and fewer clicks (2 or 3). In contrast, Edge and Vivaldi require an extra click and some cognitive work because the user must locate a less obvious icon and navigate through a small options menu. These extra steps can hinder ease of use, especially for first-time users. A and B have the best scores because they do not present the user with a popup asking them to name the workspace; they can click the button, and the window will be saved. If the user wants to name the workspace, they can do so later.
Note: Not providing a step for naming the workspace may be problematic or unwanted for some users. The user can always rename the workspace later, but this may not be obvious to everyone. This design was not tested in the evaluation, so there is no user feedback in this thesis about this change.

Prototype	STG	Edge	Vivaldi	B	A
Mental Work	0	0	0	0	0
Clicks	3	3	3	2	2

Table 8.4: Creating a window using workspaces: Comparison of mental work and click count for non-Firefox prototypes.

Creating a Window via Workspaces When creating a new window via workspaces, see table 8.4, B and A are slightly more efficient with only 2 clicks since they automatically accept default settings unless customisation is desired, while STG, Edge, and Vivaldi require 3 clicks. This demonstrates that while the basic process is simple, there is room for reducing the complexity of the initial action; however, this may

also likely decrease the likelihood of a user finding the renaming and customisation functionality.

Prototype	STG	Edge	Vivaldi	B	A
Mental Work	2	N/A	2	2	2
Clicks	2	N/A	2	2	2

Table 8.5: Resuming a window in the current window using workspaces: Comparison of mental work and click count for non-Firefox prototypes.

Resuming in the Current Window When resuming a window in the current window using workspaces, table 8.5 shows that for most prototypes (STG, Vivaldi, B, A), resuming a window in the current window requires the same amount of mental work and clicks. However, Edge does not support this approach and has a limitation in its functionality compared to the others.

Prototype	STG	Edge	Vivaldi	B	A
Mental Work	3	2	3	3	3
Clicks	3	2	3	3	3

Table 8.6: Resuming a window in a new window using workspaces: Comparison of mental work and click count for non-Firefox prototypes.

Resuming in a New Window When resuming a window in a new window using workspaces, see table 8.6, a slight variation is noted for resuming in a new window. Edge stands out by offering a quicker process (2 mental work, 2 clicks). The other prototypes (STG, Vivaldi, B, A) require marginally more effort (3 mental work, 3 clicks). This indicates that while all prototypes facilitate the task, Edge provides a more streamlined experience in this particular case, as it is the default and, in fact, the only case. In contrast, the other prototypes incorporate it as a secondary feature.

8.8 Summary of Task Analysis

The analysis shows that none of Firefox's three current methods for preserving the browsing state, keeping windows open, bookmarking all open tabs, and restoring recently closed windows provide a good solution for saving and resuming work.
Each method has drawbacks. Keeping windows open increases the cognitive workload, bookmarking requires a lot of organisation, and restoring recently closed windows is limited by a fixed number of slots. None of these features are designed for window saving, and the competitor and task analysis show that there is a clear gap in Firefox's support for seamless session management.

In addition, the current methods in Firefox are not user-friendly. Users must navigate through several menus and options to save their tabs, which is a time-consuming and frustrating process. In contrast, browsers such as STG, Edge, and Vivaldi offer a more straightforward and intuitive way to save windows. They also allow users to assign custom names to their saved windows, making it easier to find them later, a helpful feature for managing multiple windows. Firefox windows in the taskbar display the name of the last open tab, which can be challenging to navigate when multiple windows are active, as the name of the last open tab often changes frequently.
Resuming workspaces in a new window requires fewer steps for Edge than the other prototypes. However, it cannot resume a workspace in the current window, unlike all other tested prototypes. This reduces its versatility and disables some potential user workflows.
Prototypes A and B aim to minimise mental work and clicks, particularly when saving windows. The design utilises good defaults and delays optional actions, such as naming, to reduce friction for impatient users. This approach is intended to lower the barrier to entry without sacrificing flexibility. In fact, a "New Workspace" feature may be unnecessary: users already know how to open a new window; what they lack is a simple way to make it permanent. Asking them to click "New Workspace" forces a new mental model for a potentially small benefit: combine the actions of creating a new window using the browser and saving that window into a workspace. Creating a new window requires two clicks in most browsers and likely happens relatively infrequently. It is unclear how much value reducing these two clicks provides to users, so further research is required.
The analysis focused on measurable factors, such as the number of clicks and the perceived mental work, but usability is not just about numbers. How easy it is to discover these features, deal with errors like opening the wrong window, and manage multiple windows without confusion are equally important metrics. Edge and Vivaldi both have room for improvement in these regards.
Comparing the workspace solutions offered by STG, Edge, and Vivaldi to the experimental prototypes A and B, the apparent benefit is their streamlined design. These systems allow users to save windows quickly and, importantly, fully customise them later by changing their names, icons, and colours. This flexibility is valuable because it helps users stay organised without introducing too much extra work upfront.

9

Exhaustive Breakdown of Workflows

This section defines common workflows users follow when managing tasks across windows. It is important to consider not only browser UI elements but also the broader usage patterns that influence how users organise tasks across multiple windows. Users may, for example, spread tabs across different windows for logical grouping or side-by-side viewing. Previous studies have shown that users require workflow flexibility to accommodate various working styles [3], [4]. Additionally, the interview participants presented several of their own workflows, including the use of multiple monitors, multiple devices, and different workflows [**IT11**], [**IT12**], [**IT13**].

Using data modelling terms, a task performed in one window represents a one-to-one relationship [43]. A task spread over multiple windows is a one-to-many relationship, while several tasks in one window indicate a many-to-one relationship. When multiple tasks span several windows, it forms a many-to-many relationship. These categories cover all possible workflow organisations. Figure 9.1 illustrates these workflows.

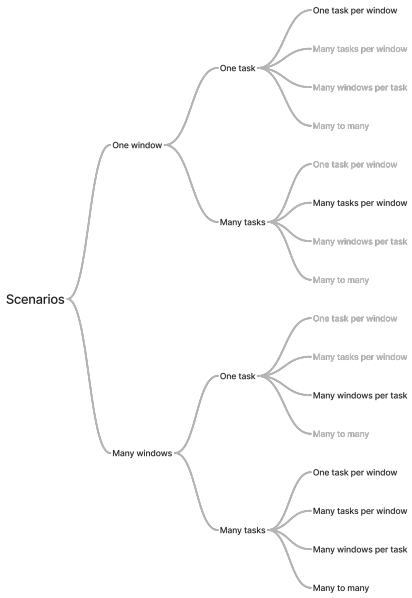


Figure 9.1: A tree diagram showcasing all possible scenarios at a low level of abstraction, the leaves of the trees represent what workflows can be achieved using windows and tabs as the only method of organisation. Leaves in grey can not be achieved by default in the described scenario.

- **One Window, One Task:** These users use a single window for one task. Many prefer this approach to minimise distractions and reduce cognitive load **tabs-and-windows**. For example, a user who works exclusively with answering emails likely has no need for multiple windows.
- **One Window, Many Tasks:** These users use a single window for multiple tasks. Some may prefer this approach when managing only a few tasks with a limited number of tabs. However, handling many tasks in one window can result in clutter and reduced usability [14]. For example, a user might open news articles, social media feeds, and streaming videos all in one window without needing to switch.
- **Many Windows, One Task:** These users work on a single task using multiple windows. This setup allows them to view more tabs simultaneously and group them logically. For example, a user working on a research report may open one window for reference documents, a second for raw datasets, and a third for drafting the report text.
- **Many Windows, Many Tasks:** These users are likely experts who manage multiple tasks across several windows. They likely use one of the four workflows described below or a combination of them.
  - **One Task Per Window:** These users manage multiple tasks, with each window dedicated to a single task. These tasks are likely organised into separate windows that remain open due to fear of losing progress. For example, a user might have one window for a research project, another

for a personal project, and a third for a work-related task. This setup allows them to keep their tasks organised and easily accessible. They are likely focusing on only one task at a time and could benefit from closing windows that are not currently in use. However, if there is no save function, they may be forced to keep all windows open, leading to clutter and increased cognitive load.

- **Many Tasks Per Window:** These users have multiple windows open and manage several tasks within a single window. Users may choose this approach intentionally, for example, when working on tasks that share overlapping tools or resources, preferring to keep everything in a single window rather than duplicating tabs across multiple windows. A user might have one window on their second screen for email, reference documents, and quick web searches, while keeping one or more windows open for one or more other tasks. If they could, they might prefer to keep their second screen window open and close all other windows except the one they are currently working on, ending up with two windows.
- **Many Windows Per Task:** These users engage in multiple tasks across several windows. This workflow lacks a straightforward mapping between windows and tasks, leading to increased cognitive load. For example, a user who needs to compare similar products side by side might open two windows next to each other to view both products simultaneously. If the user needs to do tasks like this regularly, they might end up with multiple sets of windows, two for each task. Switching between these tasks will be twice as difficult since they will need to identify which two windows belong to the same task when switching between them. This would be more challenging the more windows they have open.
- **Many to Many:** These users manage multiple tasks while using several windows per task, and some windows are shared among tasks. For example, users might have windows that belong to multiple tasks simultaneously, as well as some tasks that require multiple windows. This setup can be useful when distinct tasks share many common tabs and are also complex enough to warrant multiple windows per task. This workflow can be particularly challenging to manage, as it requires users to keep track of which windows belong to which tasks.

Cross-Device and Collaborative Workflows Workflows may also span multiple devices or involve collaboration, which brings its own set of challenges. A detailed exploration of device syncing or collaborative sharing falls outside the scope of this thesis. The next subsections offer a concise overview of these contexts.

Firefox already has a cloud-based cross-device sync system that allows users to access their open tabs on their other devices. If saved windows are implemented, research should be conducted to determine exactly how users expect and prefer their windows to transfer between devices. Precise requirements for cross-device sync are beyond the scope of this thesis.

In collaborative settings, each participant brings different browsing habits and task structures, which complicates collaboration. While special sharing permissions or

role-based organisation may be needed, a structured, shareable workspace would likely be useful to some users. However, the specific requirements for collaboration require further research.

9.1 Analysis of the Workflows

This section examines the workflows in figure 9.1 and discusses how current window-saving methods compare with potential improvements. It evaluates user needs, cognitive load, and the trade-offs involved in different approaches.

Implications for 'One Window, One Task' These users have a single window open, and they use it for a single task. For users with such simple workflows, automatic restoration of a single window is likely satisfactory. Still, an explicit save function could reassure users that their progress is preserved to address the obscure nature of automatic restoration.

Implications for 'One Window, Many Tasks' These users have a single window open, and they use this window for multiple tasks. Users who prefer to work in a single window while managing multiple tasks are likely to keep many tabs open in their window. The lack of logical separation between tasks is also likely to increase cognitive load.
Some who work like this may prefer it because they dislike having multiple windows open; a save function could allow them to start dividing their tasks into different windows and close windows not in use. This would allow them to keep only a single window open and still have multiple tasks separated logically.

Implications for 'Many Windows, One Task' These users have multiple windows open but work only on a single task. For users handling a single task across multiple windows, the default automatic restoration might be sufficient. Some of these users may prefer to keep only a single window open, but they still need to logically group their tabs within their task. If changing between saved windows remains fast and easy, these users may also consider switching to a single-window workflow. However, this specific use case is likely more appropriately solved with tab groups. In general, users who need to switch quickly between workspaces would also benefit from keyboard shortcuts to switch between saved windows and similar expert features.

Implications for 'Many Windows, Many Tasks' These users have multiple windows open and work on multiple tasks; they are likely expert users. Users managing multiple tasks across several windows can reduce clutter and cognitive load by closing windows that belong to tasks not currently being worked on. If task switching is not too frequent, the benefits of a cleaner workspace are likely to outweigh the minor inconvenience of having to reopen those windows when needed.

Implications for 'One Task Per Window' These users have multiple tasks, and each window is dedicated to a single task. A save function would allow users to close windows associated with inactive tasks, keeping only the active window open. This reduction in open windows can lower cognitive load, although it may introduce a slight inconvenience when switching tasks. Users who frequently switch tasks may prefer keeping multiple windows open, but the ability to name each window could still allow them to identify the correct one more easily.

Implications for 'Many Tasks Per Window' These users have multiple windows open and manage several tasks within a single window. A save function that allows naming windows would enable them to create their own organisational system. For example, they might save one window per task while also maintaining a more general window that relates to multiple tasks. Saving windows would allow them to open and close task-specific windows as needed without affecting the window that handles multiple tasks. Expert users, however, may require more advanced control over their multi-task windows. One potential enhancement would be to allow expert users to connect two saved windows so that opening one automatically opens the other. This way, they could open a window for a specific task and have the general-purpose window automatically open alongside it.

Implications for 'Many Windows Per Task' These users have multiple tasks, and they use multiple windows per task. A save function would allow them to close windows not related to the task they are actively working on, ensuring that any open windows are associated with the active task to reduce cognitive load. A potential challenge is that users managing many windows and tasks may accumulate a large number of saved windows. While naming each window could provide a temporary workaround by helping users group related windows in the saved list, a more effective solution is to implement a folder-based organisation system. This system would allow users to manage and organise their saved windows efficiently.

Implications for 'Many to Many' These users manage multiple tasks while maintaining multiple windows per task, with some windows shared between tasks. This workflow combines aspects of the previous cases. Expert users might prefer a system similar to that described in 9.1 that allows windows to be configured to open and close based on user-defined criteria. This would enable them to create complex workflows that automatically open and close windows based on the task they are currently working on.

Implications for 'Cross-Device and Collaborative Workflows' Users working across multiple devices can benefit from saving windows that are shared between devices, ensuring a consistent workflow regardless of their location. In collaborative settings, special solutions may be needed because each user has different habits. A folder system could help multiple users organise their windows, improve collaboration, and offer the ability to share a folder with someone, allowing all saved windows within that folder to be shared.

9.2 Summary of Workflow Analysis

This breakdown shows that user approaches range from minimal to very complex. It is clear that user workflows are diverse and that not every user will benefit from the same features. Being able to save windows is mostly useful for all workflows, even simple ones, because it reassures users that their progress is saved.
Customising window names and colours allows users to separate tasks logically and navigate more easily. Quick and efficient switching is essential for many users' workflows.
For users who frequently switch between tasks, keyboard shortcuts for switching between saved windows would be beneficial. This feature would allow them to quickly access the windows they need without having to navigate through the interface.
A folder-based system would help group and manage saved windows effectively, especially when multiple windows are used for the same task.
Folders are also useful for other applications where users want to logically group their saved windows, for example, separating work and free time or grouping related tasks.
Advanced linking features may be considered for expert users; however, further research is needed to determine if they are necessary.
Finally, cross-device synchronisation would likely be useful for most users and is technically required for collaboration. For collaboration, a folder system is likely also required, but further research is needed to determine if more advanced features are necessary.

10

Interview Themes and User Needs

This chapter presents the key findings from five semi-structured interviews and secondary research, focusing on how users manage browser windows and tabs. The first section identifies common themes from the interviews, supported by quotes from the coded transcripts. The second section translates these themes into a set of user needs. These needs reflect what users consider important in tools that support saving, restoring, and organising browser windows and form the foundation for later design and evaluation stages.
The interviews involved five participants selected through convenience sampling, more information about the participants can be found in section 6.5. Thematic analysis was used to identify recurring behaviours, challenges, and expectations related to session management.

10.1 Interview Themes

This section summarises the main themes from the interviews. It highlights common behaviours and challenges users face when managing their browser windows and tabs, illustrated by quotes and coded data from the interviews.

[**IT1**] **Users keep windows and tabs open to re-access tabs**
A.2.3 A.2.2 A.2.5
"I mean, of course I'm more scared to close Windows because that implies they need to go through all of the windows. All of the tabs to see if it's still important or not." -p1

[**IT2**] **Users would close windows more often if they could save them**
A.4.7 A.4.9
"While I sometimes have several windows open because I am in the middle of something, I want to close them to feel like - no, now I'm done and I don't want things that stress me out around. Then, I want to be able to open exactly the same tabs again, perhaps the next day, to continue where I left off." -p4

[**IT3**] **Users are afraid of losing their tabs**
A.3.1 A.6.3
"I am constantly working with a kind of underlying stress. If I have many important tabs open, I feel this background stress. I worry that if my computer crashes now or if I close Firefox now, I would be screwed." -p4

[**IT4**] **Users have difficulty locating specific tabs and windows**

A.1.1 A.1.2 A.1.6
"Yes, it's like I know it's somewhere on this screen, and then I just have to scroll and search, you know." -p3

[**IT5**] **Users identify windows by hovering to preview them and through trial and error**
A.1.6 A.1.9
"But if I don't have it, then I usually just hover and look at what seems most right. Sometimes they almost look the same. Then you just have to try opening one of them and see, is it the right one or not?" -p4

[**IT6**] **Users identify tabs by their icons, names, location, hovering to preview, and through trial and error**
A.1.3 A.1.5 A.1.4 A.1.8
"I mostly go by color. I know that Overleaf is green, so I find it easily, and like what position it has in my window, by place and color." -p2

[**IT7**] **User would like to name their tabs and windows**
A.4.2 A.4.1
"Yes, I myself would probably have just given them a short name that I know, so I know what the page is about, like if I'm writing a report on AI, then I would have written AI 1, and then if I get another one about AI, I might have named it AI 2, and that feels easier to remember than just having no name at all." -p3

[**IT8**] **Users do not want to save every tab**
A.9.1 A.9.9 A.5.6
"I do that, but then I usually open them, and it's kind of like a reset there. So then I open, maybe the next day, just those in the order that I actually need to do things. So then I don't have 20 tabs open. But I will have that many by the end of the day." -p4

[**IT9**] **Users want to keep their windows tidy, either by manually cleaning them or by occasionally recreating their workflow**
A.9.1 A.9.9
"Every now and again, I do a check and close the tabs that I feel like, ah, I haven't looked at this one for a while. And just clean up a bit so that I actually know what different tabs I have in my various windows." -p4

[**IT10**] **Users forget what tabs are in their saved windows after some time**
A.6.1
"But then, when you go back to a [window] that you haven't used or been in for a long time, then it's just - I have no idea. I don't remember what stuff is in here, you know." -p5

[**IT11**] **Users have different preferred workflows**
A.8 A.9
"Uh, it's it gets messy really easily. I feel like ideally I would like to have them. One project. In one window, but right now, for example, my computer and now I have six different windows open with at least 15 tabs in each, and they're all mixed and I just kinda I need to have this mental model inside my

head to remember which one is which." -p1
"No, I wouldn't say that I have different windows open at the same time. If anything, I have them on different screens, but on each screen, I usually have just one window open." -p3
"So, I always try to, if there's a window with a tab related to a course, then I often want to drag that tab over to the correct window where all the other tabs for the same course are." -p5

[**IT12**] **Users often use more than one screen**
A.9.6
"I use ChatGPT a lot, so I always have it on the left screen, like the whole screen is just ChatGPT. Then, if I'm writing something, I have it on the middle screen. And then, if there's something I need when I'm writing, maybe some info text or something, then I have that on the right screen..." -p3

[**IT13**] **Users operate multiple devices, and they likely want to share information between them**
A.9.5
"It's two different computers. Usually, anyway. Sometimes, like if I'm working from home, for example, after I finish work, I can also use my work computer for some leisure activities." -p4

[**IT14**] **Users are more or less organised depending on the task**
A.7
"Yes, it also depends a bit. So, whether I am working or just leisurely browsing, there is quite a big difference." -p4

[**IT15**] **Users often do not plan their tasks in advance, and their tasks grow over time**
A.9.2 A.9.4 A.9.11 A.9.13
"And I realised that [the tasks] are evolving because. Maybe at the beginning I will have everything in the design [window]. But as more they grow. Yeah, as more I will need to kind of separate them." -p1

[**IT16**] **Users do not want to be forced to organise**
A.9.8 A.4.2
"But like, this thing about being able to name things and stuff like that. That would have probably bothered me at first, because I feel. Yes, because I feel I haven't needed to do that before. Why should I need to do it now?" -p2

[**IT17**] **Users want a simple and efficient method to save their windows**
A.4.5
"Absolutely, especially if it was very very easy, just a small thing like this: save." -p2

10.2 User Needs

Building on the interview themes and secondary research, this section outlines the main user needs. Each need is directly grounded in multiple participant com-

ments—that any solution should satisfy and that guide the design by reflecting what users find most important for managing their tabs and windows.

[**UN1**] **State Preservation**
[**IT1**] [**IT2**] [**IT3**] [3]
Users need to be able to save and close their windows without losing the tabs they contain or their context.

[**UN2**] **Peace of Mind**
[**IT3**]
Users need to feel that their tabs and windows are safe and easy to recover.

[**UN3**] **Ease of Use**
[**IT17**]
Users need a simple and efficient interface and functionality to minimise their learning curve and effort.

[**UN4**] **Visual Cues**
[**IT4**] [**IT5**] [**IT6**] [3] [9]
Users need to easily locate and reopen specific saved windows and tabs using previews, icons, names, colours, and other distinguishing features.

[**UN5**] **Flexible Workflows**
[**IT11**] [**IT12**] [**IT13**] [3] [4]
Users need flexibility to organise tabs and windows in ways that match their workflow and preferences.

[**UN6**] **Unstructured Work Support**
[**IT11**] [**IT14**] [**IT15**] [4]
Users need to save and reorganise tabs and windows on the fly as their tasks emerge and shift.

[**UN7**] **Non-intrusive Organisation**
[**IT16**] [**IT14**] [**IT17**]
Users need organisational features that do not force extra work or attention but help keep their workspace tidy.

[**UN8**] **Tidy Workspace**
[**IT8**] [**IT10**] [**IT9**] [**IT11**] [14]
Users need the ability to maintain a tidy workspace.

[**UN9**] **Overview and Reminders**
[**IT10**] [3]
Users need reminders, summaries, or overviews of saved tabs so they remember what is in their saved windows over time.

[**UN10**] **Cross-Device Sync**
[**IT13**]
Users need seamless syncing and sharing of saved tabs and windows across their devices to maintain continuity.

10.3 Analysis of User Needs

State Preservation Users want to save their work without added complexity. Saving and restoring tabs and windows should be easy and reliable. When they reopen a window, it should look and feel exactly as it did before. The system must preserve the order of tabs, which tab was active at closing time, which tabs were pinned in the window, the scroll position on each page, the navigation history used for back and forward navigation, and any other session details. Preserving these details intact provides users with context and continuity, allowing them to resume their tasks more effectively.

Peace of Mind It is essential to ensure that users feel secure, knowing they will not lose tabs or windows when using the system and can easily recover them if they do. This could be achieved by providing visual confirmation when saving windows, reassuring users that their work is safe. For example, a small popup message could appear when a window is saved, confirming that the action was successful. However, even if the system worked perfectly and never lost any tabs or windows in the first place, there is still a risk of user error. Being able to restore lost windows, whether due to an accident or a crash, is essential. This could be achieved by providing a history of saved windows, allowing users to restore them, perhaps even if they were not saved in the first place. This would give users peace of mind and allow them to focus on their work without worrying about losing their tabs or windows.

Ease of Use Users need an interface that works immediately, with no steep learning curve. Key actions such as saving and restoring windows should be simple and intuitive.

Visual Cues Visual elements, such as previews, icons, and names, help users quickly identify and navigate tabs and windows. Making these visuals prominent and allowing users to customise them could significantly enhance the overall experience.

Flexible Workflows The interface should be adaptable to the various ways people work. Users often have multiple screens or devices; some prefer to use many windows, while others use just one window at a time. By avoiding assuming a single approach and instead designing a flexible system, users can do what works best for them. One way of achieving this is to allow users to change their settings to adjust the default behaviour when opening a workspace. That way, users can choose whether a new window opens as a blank workspace or as a copy of the current workspace.

Unstructured Work Support The tool should support flexible workflows by allowing users to easily rearrange tabs and create or adjust workspaces. Often, users open multiple tabs in a window and later decide that those tabs belong in a separate workspace. Therefore, it must be simple to move tabs between different workspaces. Users might also open a new window and later decide to save it as a workspace. Creating a workspace does not always happen first. New users most likely start

with open windows; because of this, rather than creating a fresh workspace upfront, they are more likely to want to save their existing windows. This implies that the functionalities of creating new workspaces and saving existing windows, two sides of the same coin, should be equal. If anything, saving windows should be prioritised since first-time users will likely find it the most useful.

Non-intrusive Organisation It is important to support self-organisation without forcing it on the user. For example, requiring users to name windows when saving them should be optional rather than mandatory. Even asking the user to name their window might be unnecessary; in the experimental prototypes, the default behaviour is to name the window based on the content of the last open tab. Whether users should be prompted to name their window while saving is debatable and requires testing. Either way, a saved but unnamed window should come with good defaults for window names and icons. This way, users can still have a tidy workspace without extra work.

Tidy Workspace Users want to keep their workspace tidy. While closing tabs currently supports this, automated solutions could provide helpful support for this activity as long as users still feel they have control over the situation and are not at risk of losing their work.

Overview and Reminders Mechanisms such as reminders, summaries, or overviews of saved content can help users recall what they have saved. One example of this would be to provide a button for viewing all tabs in the current window in full screen. Firefox already has a similar feature, but the provided interface is a small popup, and tabs are not accompanied by tab previews.

Cross-Device Sync Syncing and sharing workspaces seamlessly across multiple devices is likely an essential feature for some users. However, since some users prefer to keep their devices separate, making this feature optional is important.

11

Comparative Usability Test

This chapter presents the key findings from the comparative usability test. The test aimed to identify which methods for switching between windows and saving windows felt most natural to users, and to uncover strengths and weaknesses in competing designs.

Three interactive mockups were tested: two based on the designs used in Microsoft Edge and Vivaldi, and one with alternative features created for this study. All mockups followed Firefox's visual design to reduce bias and help users focus on the interaction.

The same 5 participants described in section 6.5 used each mockup to perform common tasks, such as saving a window or switching between workspaces. User feedback and observations were grouped into themes and analysed. The following list summarizes the main findings derived from this thematic analysis.

[**CU1**] Users experience significant problems when trying to save their current windows in both Edge and Vivaldi browsers

[**CU2**] Some users prefer naming their windows directly when they save them

[**CU3**] Users appreciate the ability to choose both colour and icon for their windows

[**CU4**] Users do not appreciate the extra action buttons that appear when a saved window is selected, like in Edge

[**CU5**] Users seem to understand that they can reorder windows by dragging, even without visible drag handles

[**CU6**] Users seem to find the triple dot menu clearer than using right-click to open the context menu

[**CU7**] Some users naturally double-click to rename a saved window

[**CU8**] Users seem to prefer new windows opening only when taking an explicit action to do so, unlike in Edge

[**CU9**] Some users appreciate the feature that allows switching between saved and unsaved windows, like in Vivaldi

[**CU10**] Users prefer seeing their unsaved window in the interface as it helps them locate themselves more easily, like in Vivaldi

[**CU11**] Some users were slightly confused by the presence of an unsaved window when there are no unsaved tabs, like in Vivaldi

[**CU12**] Users find it easier to identify the unsaved window when it is placed at the top of the interface, like in Vivaldi

[**CU13**] Some users find it useful to open a new temporary tab through an empty unsaved window, like in Vivaldi

[**CU14**] Users are generally confident that unsaved tabs are not lost when their windows are replaced by another saved window, like in Vivaldi

11.1 Analysis of Usability Test Results

Saving Windows Users struggle with saving windows in Edge and Vivaldi because the process is unclear. Many users found the functionality of creating new empty workspaces but had difficulty saving their current windows.

Window Customization Users appreciate being able to customize windows with different colours and icons. This personalization might help them feel a sense of ownership and make it easier to quickly identify and organize their workspaces.

Action Buttons and Context Menus Users do not appreciate explicit action buttons, at least not how they appear in Edge. They overload the interface visually and distract users from what matters, though some might find them helpful, especially if they are not familiar with the interface. Further research is needed to determine this.

Drag-and-Drop and Inline Renaming Natural interactions such as dragging windows to reorder or double-clicking to rename them work well because they match familiar actions that users have encountered in other programs. Users are comfortable with these interactions, which means they can be used to improve the interface's usability. However, it is important to note that not all users are comfortable with these features, so clear visual cues or instructions may still be necessary. For example, some users prefer the triple dot menu over the right-click context menu. The triple dot icon is more visible and immediately suggests additional options, making it easier for users to find and use the features they need.

Opening and Replacing Windows Regarding opening workspaces in new windows as opposed to replacing the current window, users seem to only like them to open only when explicitly triggered. This is likely because a deliberate action gives them better control. However, replacing the current window might make users feel like they have accidentally lost their work, though the test participants did not feel this way. It is unclear whether this is a problem with the design of the interface or if it is simply a matter of user preference.

Managing Unsaved Windows If unsaved windows are to be visible inside the interface, they should be placed so that it is easy to identify and understand what it is. It should most likely be placed at the top of the interface; the test participants preferred this option. However, when no unsaved tabs are present, having an unsaved window visible can be confusing. This suggests that the interface could improve by adapting the display based on whether unsaved work exists.

12
Evaluation

This section reports the results of an evaluation in which participants 59 participants were evenly assigned to one of four prototypes (Prototypes A, B, Edge, Vivaldi) and asked to complete five representative tasks: saving a window on the first and second attempts, switching between an unsaved and a saved window, switching between two saved windows, and opening a saved window in a new window (the last task did not apply to Edge, since workspaces always open in a new window). For each task, users rated how easily they completed the task and how confident they were that they succeeded without issues. Objective misclick counts and completion times were also recorded. Participant demographics were also recorded, along with prior experience with window-saving tools. The test ended with a SUS (System Usability Scale) test to measure overall satisfaction and learnability across the four prototypes.

12.1 Demographics and Metadata

Participants were evenly distributed across the four prototypes, with each group comprising roughly the same number of users; see figure 12.1. Age profiles were also similar: most participants fell within the 18-34 year age range, and no prototype deviated significantly from this range, see figure 12.3.
The gender breakdown showed no statistically significant differences overall; see figure 12.2. However, Prototype A and Vivaldi each had nearly equal numbers of male and female participants, while Prototype B and Edge had a higher proportion of male users. Browser preferences varied too; see figure 12.4, with Prototype A notably having far more Safari users than expected.
Prior experience with dedicated window-saving tools differed across all prototypes. While the majority in each group had never used such a feature, see figure 12.5, Prototype B had the highest share of first-time users. Prototype A stood out for having more participants who rely on bookmarking or other workarounds. Conversely, Vivaldi and Edge recruited more users who regularly use a built-in window-saving feature.
Self-rated computer skills likewise did not differ significantly by prototype, see figure 12.6. Still, Prototypes A and Vivaldi had more participants who judged themselves "average," while B and Edge included more who considered themselves "very skilled."

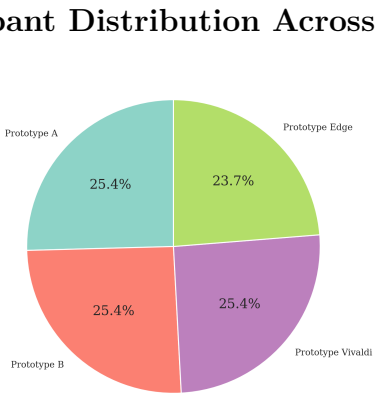


Figure 12.1: Number of participants across dataset prototypes. Distribution is nearly equal across all prototypes. Edge has 14 participants, Vivaldi has 15, Prototype A has 15, and Prototype B has 15.

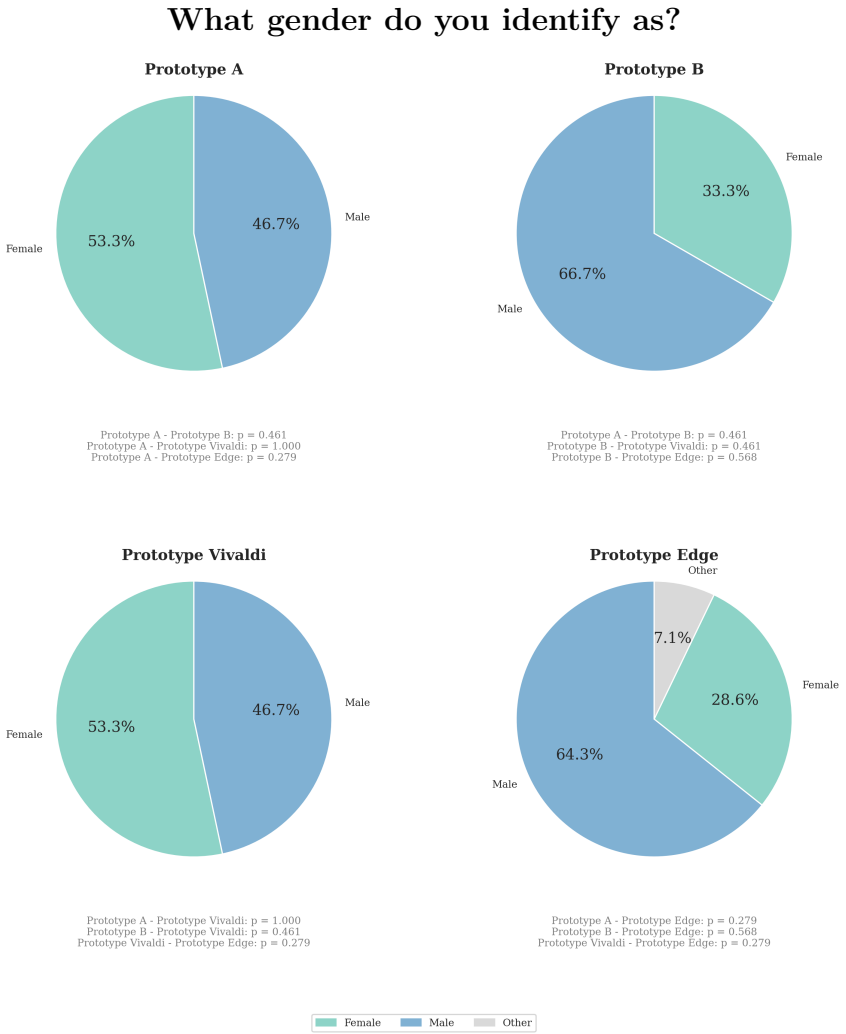


Figure 12.2: Distribution of participant genders for each prototype. Chi-Squared Test of Independence did not reveal any significant differences.

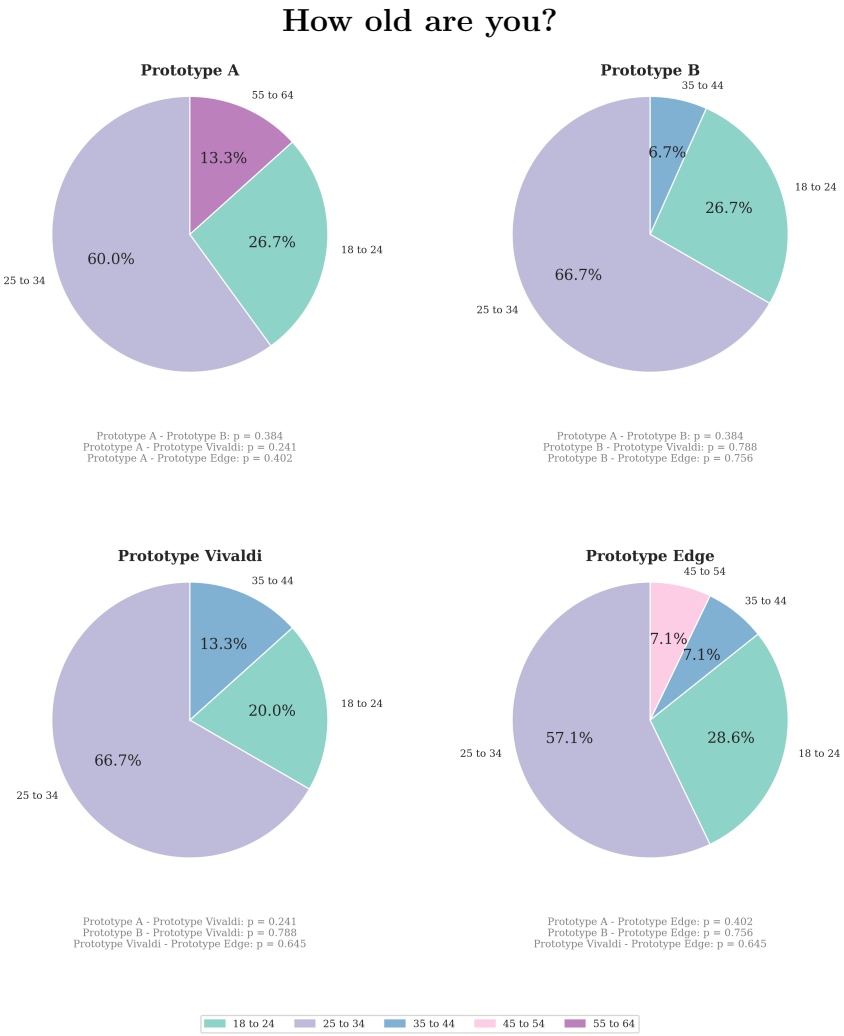


Figure 12.3: Distribution of participant ages for each prototype. Chi-Squared Test of Independence did not reveal any significant differences.

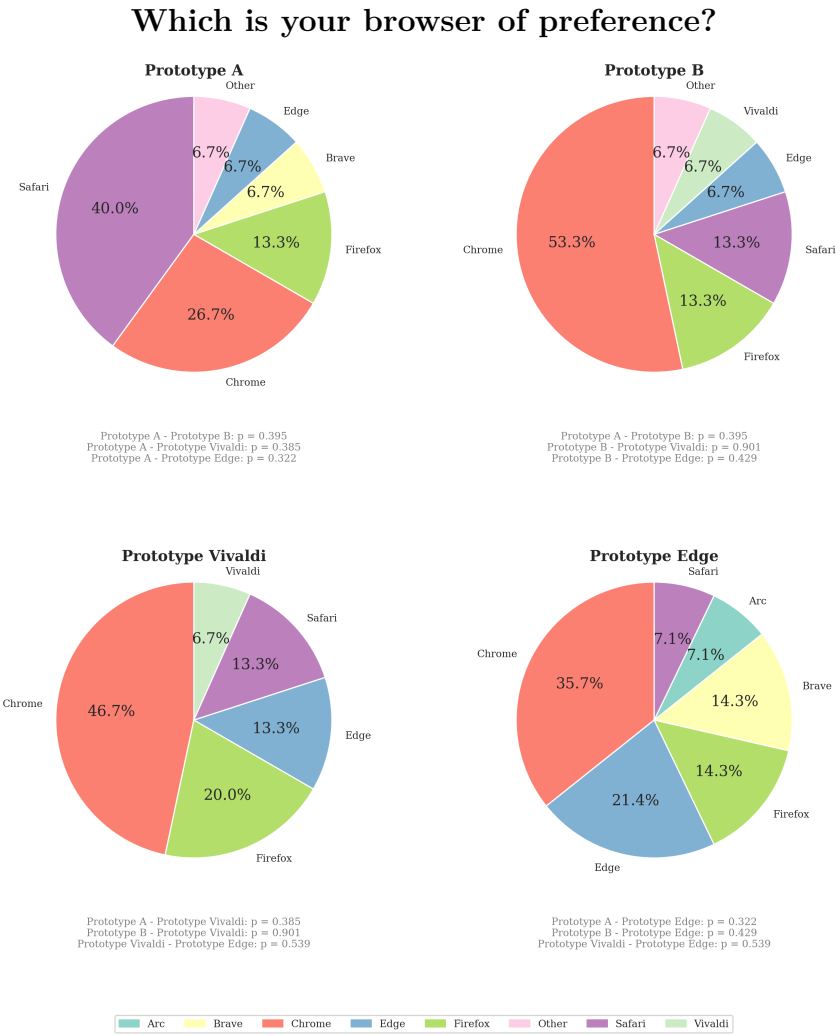


Figure 12.4: Distribution of preferred web browsers among the participants for each prototype. Chi-Squared Test of Independence did not reveal any significant differences.

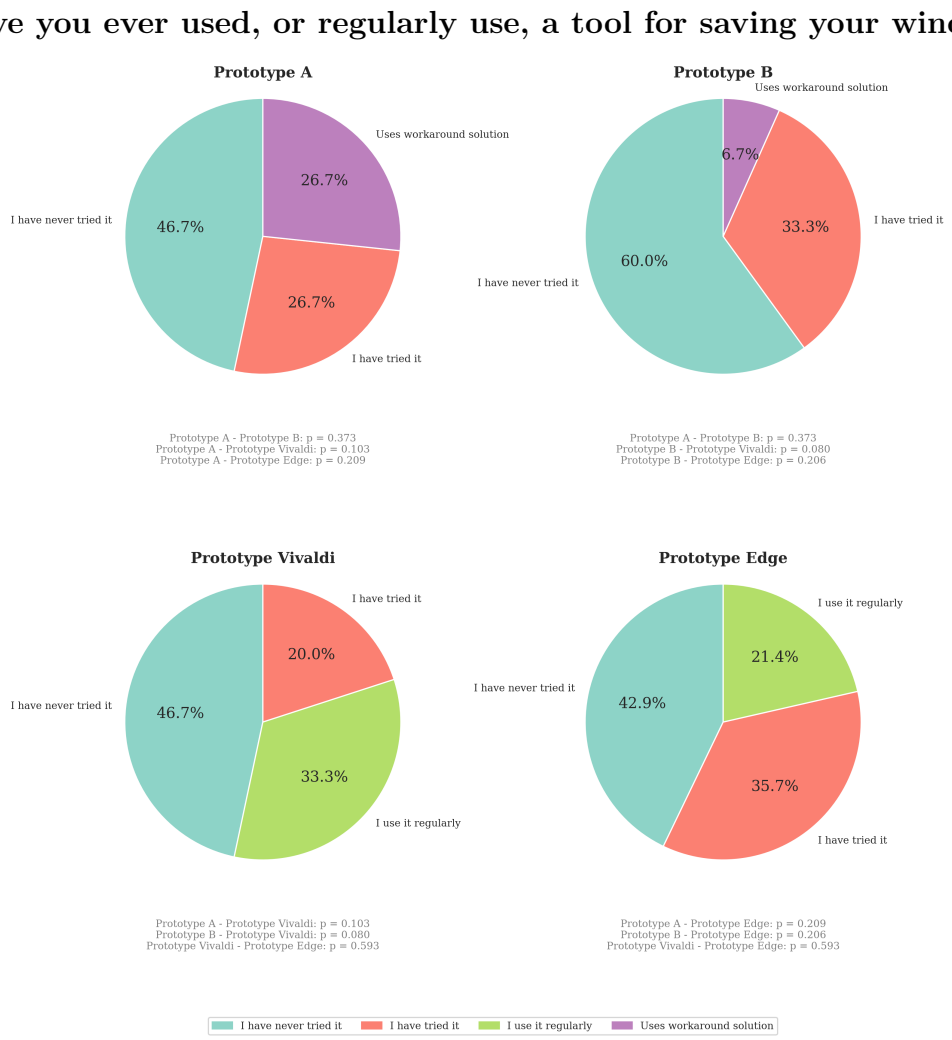


Figure 12.5: Previous experience with window-saving tools among the participants. Chi-Squared Test of Independence did not reveal any significant differences.

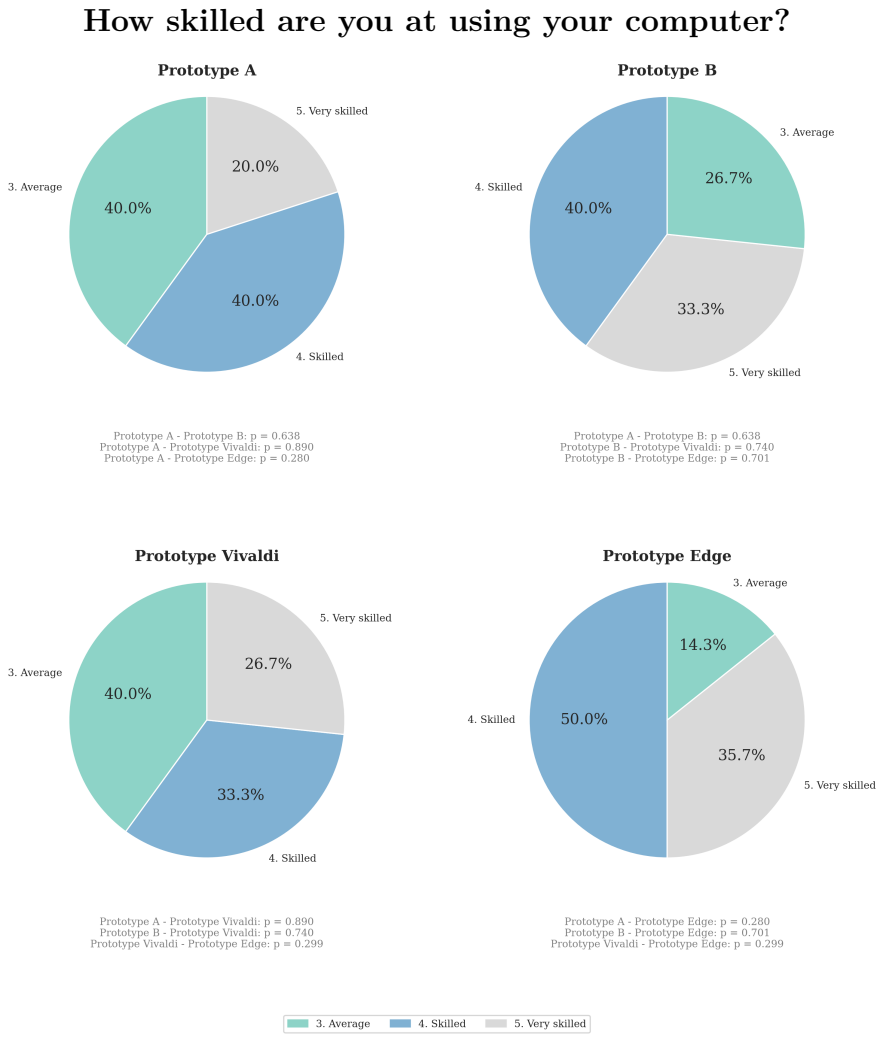


Figure 12.6: Self-rated computer skill levels of the participants for each prototype. Chi-Squared Test of Independence did not reveal any significant differences.

12.2 Aggregate Usability Results

Figure 12.7 shows overall usability as measured by the SUS. Prototype B scored significantly higher than Vivaldi.
Figure 12.8 summarises average task ratings (excluding Task 5 and SUS). Both Prototypes A and B received significantly higher ratings than Edge and Vivaldi.
In figure 12.9, average misclick rates are displayed. The overall test did not find a clear difference among prototypes. Still, one pairwise comparison, Prototype B versus Vivaldi, did reach significance. Since the overall test was not significant, this single pairwise finding should be interpreted with caution.
Figure 12.10 presents average completion times (excluding Task 5). Prototype B enabled significantly faster performance than both Edge and Vivaldi; however, the comparisons among the other prototypes were not statistically significant.

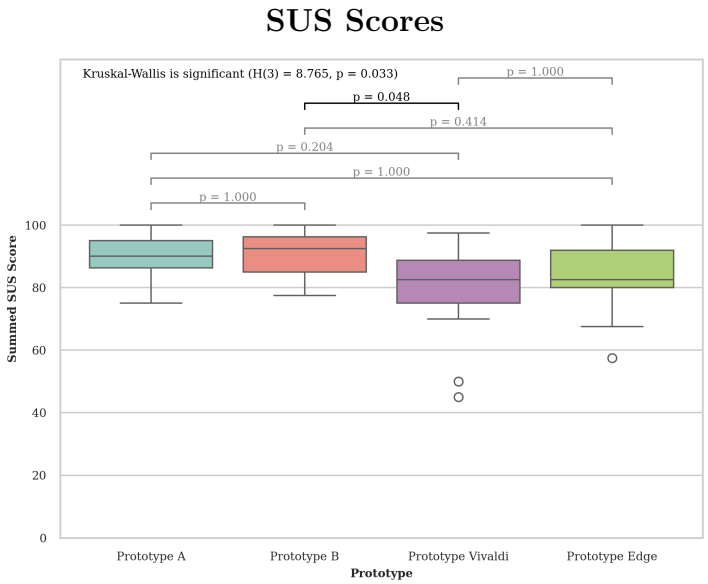


Figure 12.7: The calculated SUS scores of each prototype were analysed using a Kruskal-Wallis test (H(3)=8.765, p=0.033), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype B - Prototype Vivaldi (p=0.048); the remaining comparisons were not significant.

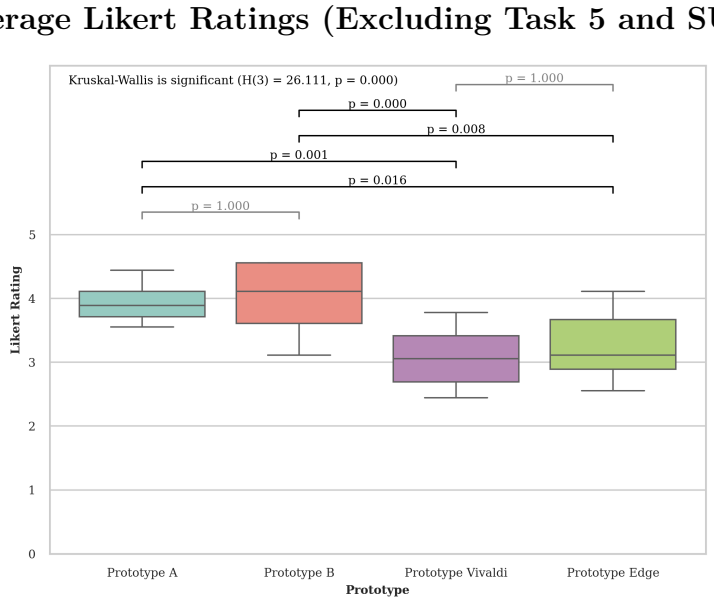


Figure 12.8: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=26.111, p<0.001), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype A - Prototype Edge (p=0.016), Prototype A - Prototype Vivaldi (p<0.001), Prototype B - Prototype Edge (p=0.008), and Prototype B - Prototype Vivaldi (p<0.001); the remaining comparisons were not significant.

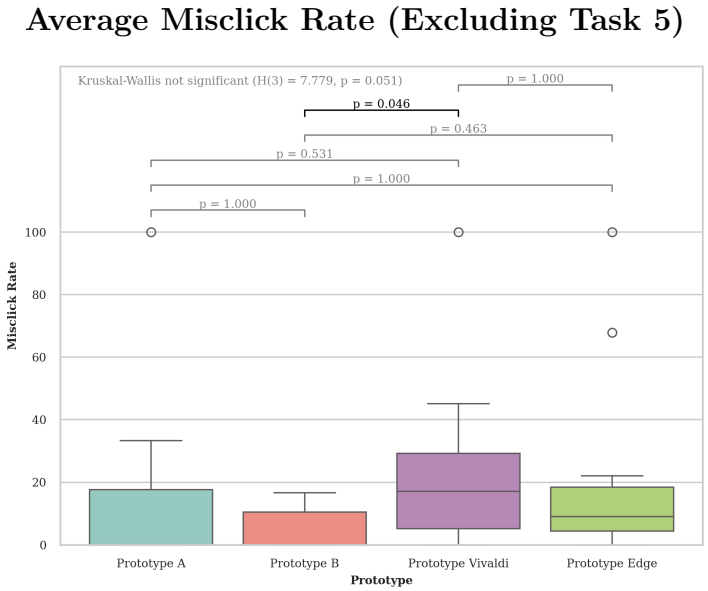


Figure 12.9: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=7.779, p=0.051), indicating no significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype B - Prototype Vivaldi (p=0.046); the remaining comparisons were not significant.

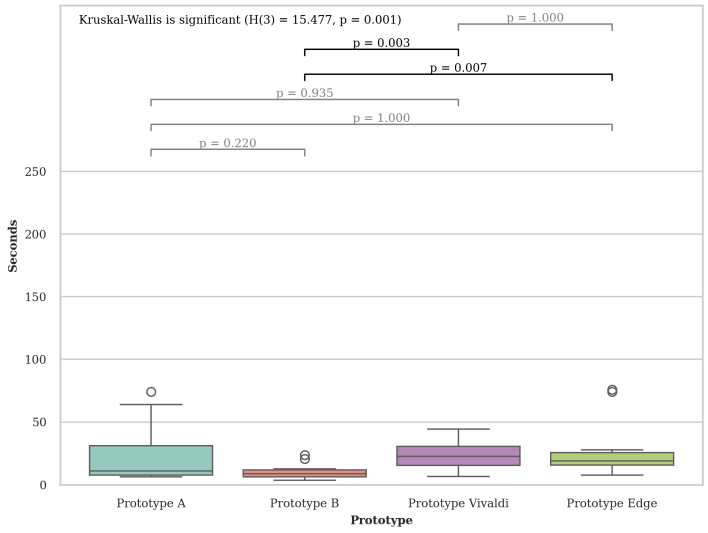


Figure 12.10: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=15.477, p=0.001), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype B - Prototype Edge (p=0.007), and Prototype B - Prototype Vivaldi (p=0.003); the remaining comparisons were not significant.

12.3 Task 1 - Saving a window on the first attempt

Participants were asked to locate and use the save-window function to save an already open window immediately after beginning their session. They then indicated how easy it was to find the feature and how confident they felt that the window had been saved.
Figure 12.11 shows that users found it easier to locate the save function in Prototypes A and B than in Edge or Vivaldi; these differences for both A and B were statistically significant.
In figure 12.12, participants reported higher confidence that their window had been saved when using Prototypes A and B compared to Vivaldi. Both A and B differed significantly from Vivaldi, while comparisons with Edge were not significant.
Figure 12.13 illustrates that Prototype B yielded significantly fewer misclicks than Edge and Vivaldi. Prototype A also had fewer misclicks than those browsers, but those differences did not reach significance.
Finally, figure 12.14 indicates that users completed the save task significantly faster in Prototype B than in Edge or Vivaldi. Prototype A showed faster completion times than both, though those differences were not statistically significant.

Task 1 - Finding how to save the window was easy for me.

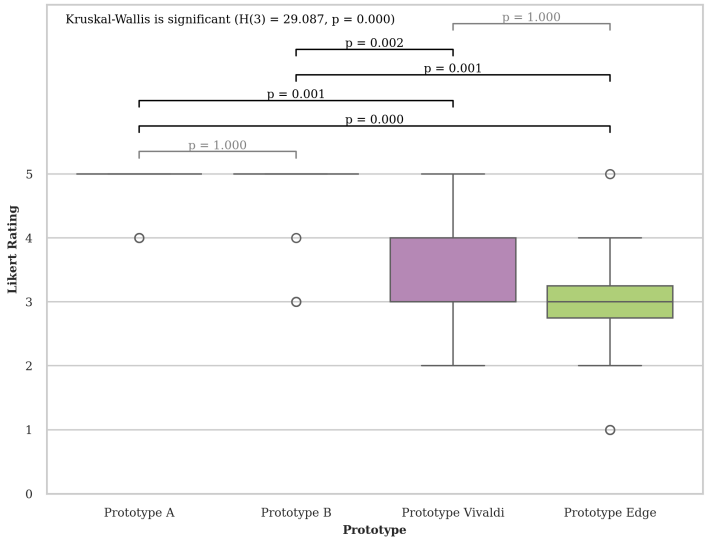


Figure 12.11: Likert scale ratings were analyzed using a Kruskal-Wallis test ($H(3)=29.087$, $p<0.001$), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype A - Prototype Edge ($p<0.001$), Prototype A - Prototype Vivaldi ($p=0.001$), Prototype B - Prototype Edge ($p<0.001$), and Prototype B - Prototype Vivaldi ($p=0.002$); the remaining comparisons were not significant.

Task 1 - I feel confident that my window was saved.

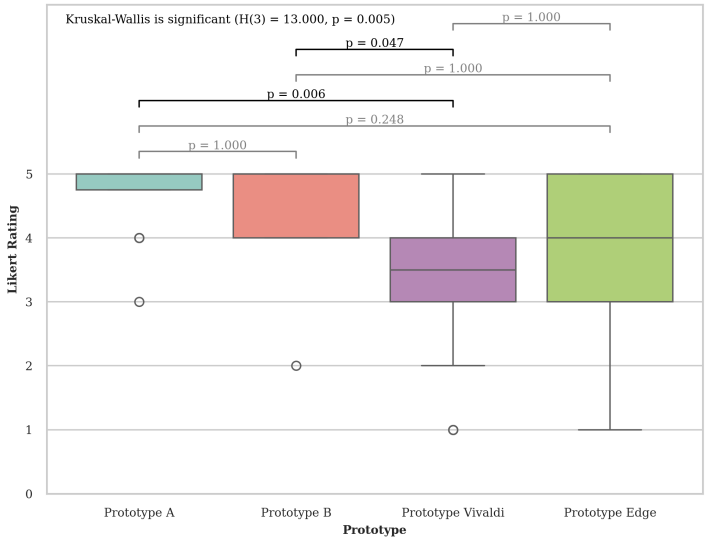


Figure 12.12: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=13.000, p=0.005), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype A - Prototype Vivaldi (p=0.006), and Prototype B - Prototype Vivaldi (p=0.047); the remaining comparisons were not significant.

Task 1 - Task Misclick Rate

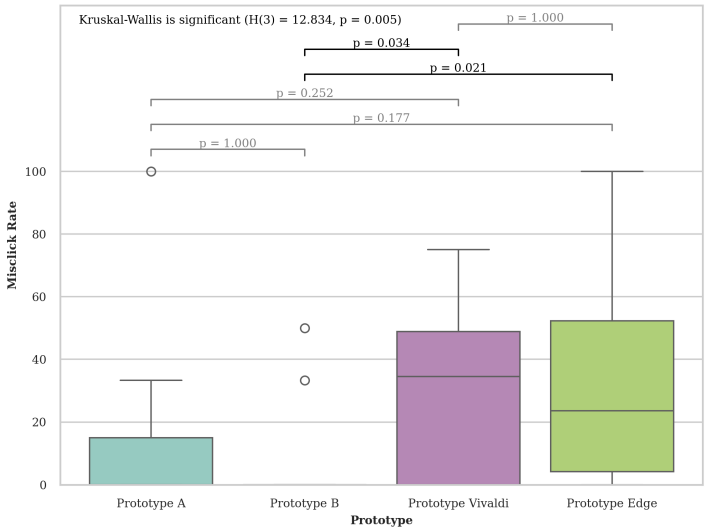


Figure 12.13: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=12.834, p=0.005), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype B - Prototype Edge (p=0.021), and Prototype B - Prototype Vivaldi (p=0.034); the remaining comparisons were not significant.

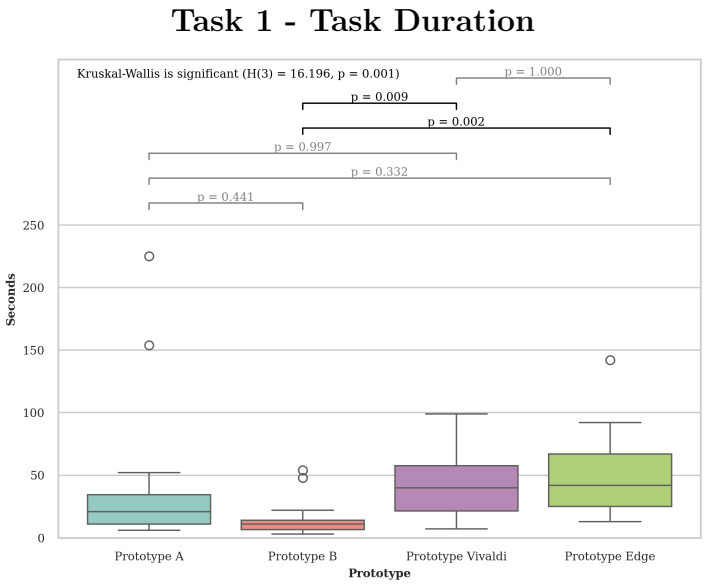


Figure 12.14: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=16.196, p=0.001), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype B - Prototype Edge (p=0.002), and Prototype B - Prototype Vivaldi (p=0.009); the remaining comparisons were not significant.

12.4 Task 2 - Switching between an unsaved and a previously saved window

Users switched back and forth between a new (unsaved) window and the window they had just saved in the previous task. They rated whether they felt any tabs might have been lost during the switch and whether they liked the way the window was switched. Depending on which prototype they were using, the question was phrased differently. Participants using Vivaldi or Prototype A were asked: "I liked that my videos opened in the current window - instead of in a new window." Participants using Edge and Prototype A were asked: "I liked that my videos opened in a new window - instead of in the current window."
In figure 12.15, participants rated how much they felt their tabs might have been lost when switching windows. Ratings were similar across all four prototypes, with no statistically significant differences.
Figure 12.16 shows participants preference for opening videos in the current window versus a new one. Scores for Prototypes A and B were slightly lower than for Edge and Vivaldi, but none of these differences reached statistical significance.
Figure 12.17 reports task misclick rates for switching back to a saved window. All prototypes had nearly equal misclick rates with no significant differences.
Finally, figure 12.18 presents the time taken to switch windows. Completion times were similar across Prototypes A, B, Edge, and Vivaldi, though slightly higher for A. Again, no statistically significant differences were observed.

Task 2 - I feel like some of my tabs may have been lost.

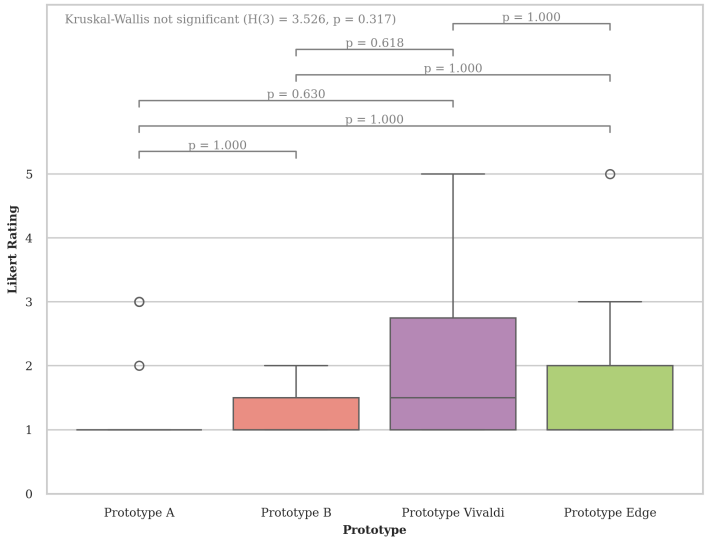


Figure 12.15: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=3.526, p=0.317), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

Task 2 - I liked that my videos opened in the current window - instead of in a new window. (alternatively: I liked that my videos opened in a new window - instead of in the current window.)

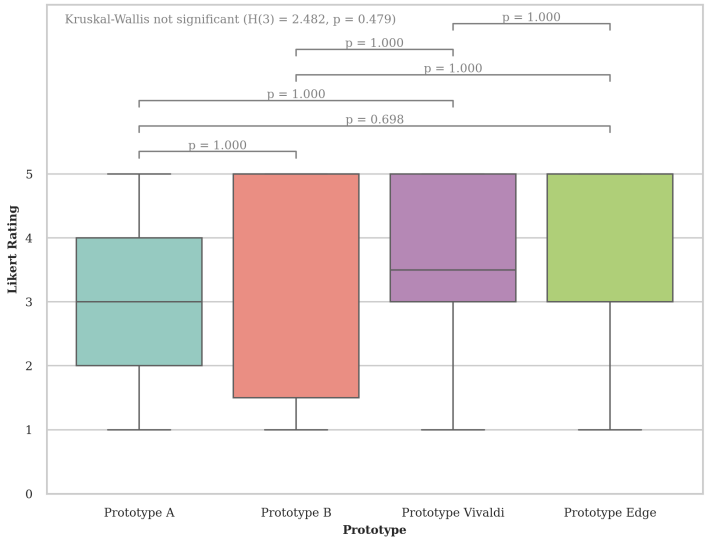


Figure 12.16: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=2.482, p=0.479), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

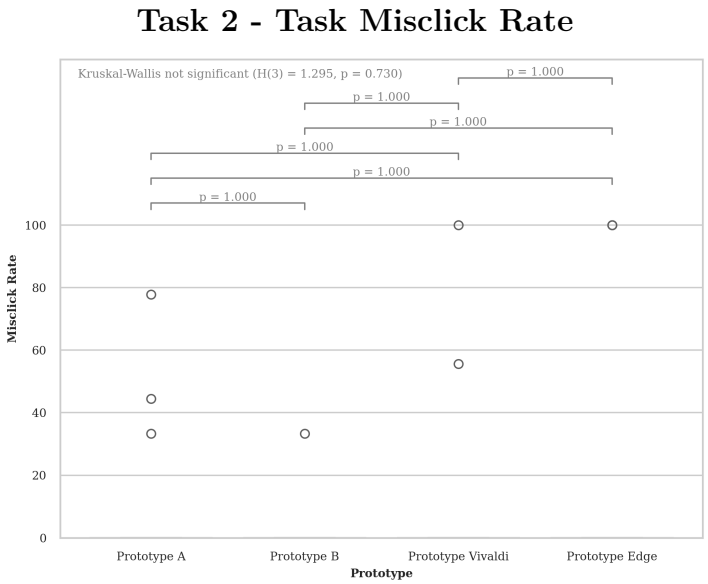


Figure 12.17: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=1.295, p=0.730), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

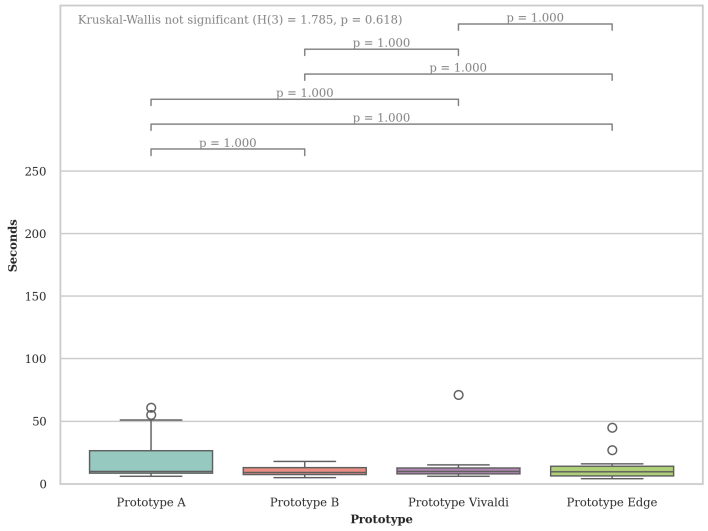


Figure 12.18: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=1.785, p=0.618), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

12.5 Task 3 - Saving a window on the second attempt

After having saved once, participants repeated the save-window action on the unsaved window from the previous task. They then rated ease of finding the save

function, confidence in having successfully saved their window, clarity of the default window names and icons, and their desire to change those defaults.
In figure 12.19, users rated how easy it was to find the save function on their second attempt. Ratings for Prototypes A and B were significantly higher than those for Vivaldi. While Edge performed slightly worse than both A and B, the difference was not significant.
Figure 12.20 shows users confidence that their window had been saved. Confidence ratings were similar across all four prototypes, with no significant differences. Vivaldi, however, was rated lower than the other prototypes.
In figure 12.21, participants judged the default window names. Prototypes A and B received significantly higher ratings than both Edge and Vivaldi. Vivaldi was rated higher than Edge, but not significantly.
Figure 12.22 presents ratings for the default window icons. Again, Prototypes A and B scored significantly better than Edge and Vivaldi. Vivaldi was also rated higher than Edge, but not significantly.
Figure 12.23 summarises users' preferences for changing names and/or icons. Users of A and B were significantly more likely to be satisfied with the defaults, while a large majority of Edge and Vivaldi users wanted to change both. It is worth noting that Vivaldi performed slightly better than Edge, although not significantly.
Figure 12.24 reports misclick rates for the second save. Misclick rates did not differ significantly among the prototypes. Vivaldi did seemingly perform worse, though not significantly.
Finally, figure 12.25 shows task completion times. Prototype B had significantly faster completion compared to Edge and Vivaldi.

Task 3 - Finding how to save the window was easy for me.

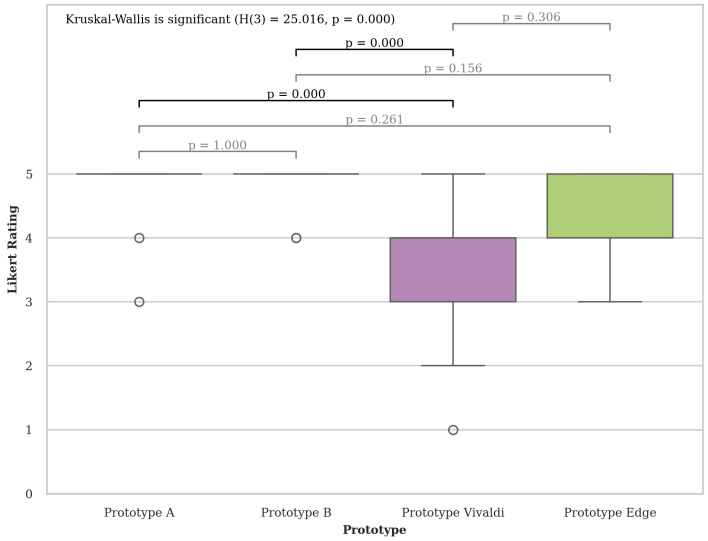


Figure 12.19: Likert scale ratings were analyzed using a Kruskal-Wallis test ($H(3)=25.016$, $p<0.001$), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype A - Prototype Vivaldi ($p<0.001$), and Prototype B - Prototype Vivaldi ($p<0.001$); the remaining comparisons were not significant.

Task 3 - I feel confident that my window was saved.

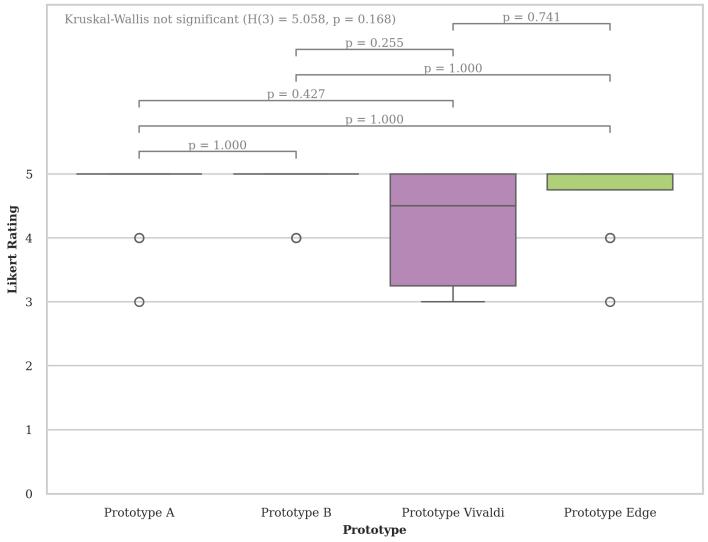


Figure 12.20: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=5.058, p=0.168), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

Task 3 - I think these names make it easy to understand which window is related to "videos" and which is related to "vacation".

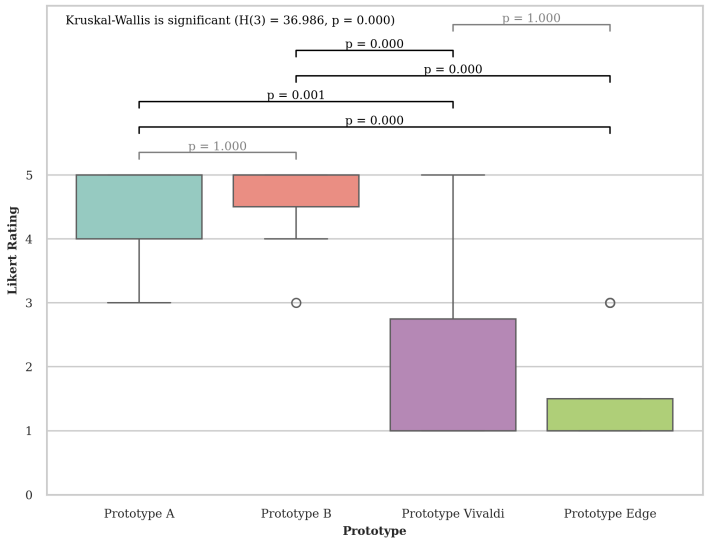


Figure 12.21: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=36.986, p<0.001), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype A - Prototype Edge (p<0.001), Prototype A - Prototype Vivaldi (p=0.001), Prototype B - Prototype Edge (p<0.001), and Prototype B - Prototype Vivaldi (p<0.001); the remaining comparisons were not significant.

Task 3 - I think these icons make it easy to understand which window is related to "videos" and which is related to "vacation".

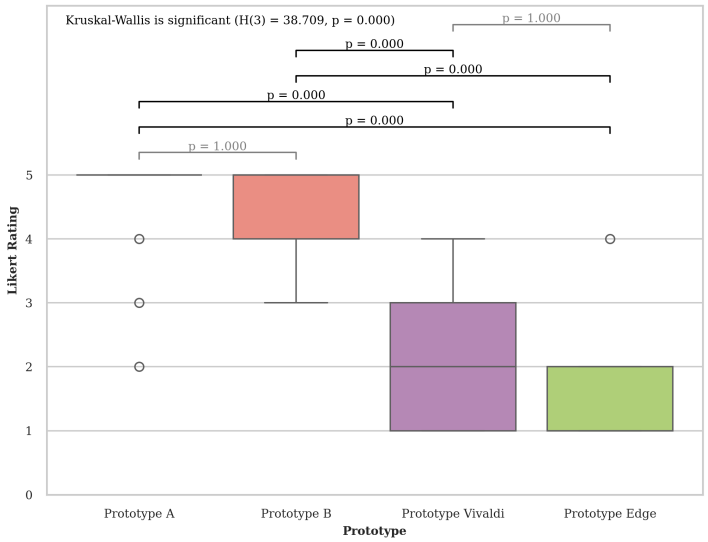


Figure 12.22: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=38.709, p<0.001), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype A - Prototype Edge (p<0.001), Prototype A - Prototype Vivaldi (p<0.001), Prototype B - Prototype Edge (p<0.001), and Prototype B - Prototype Vivaldi (p<0.001); the remaining comparisons were not significant.

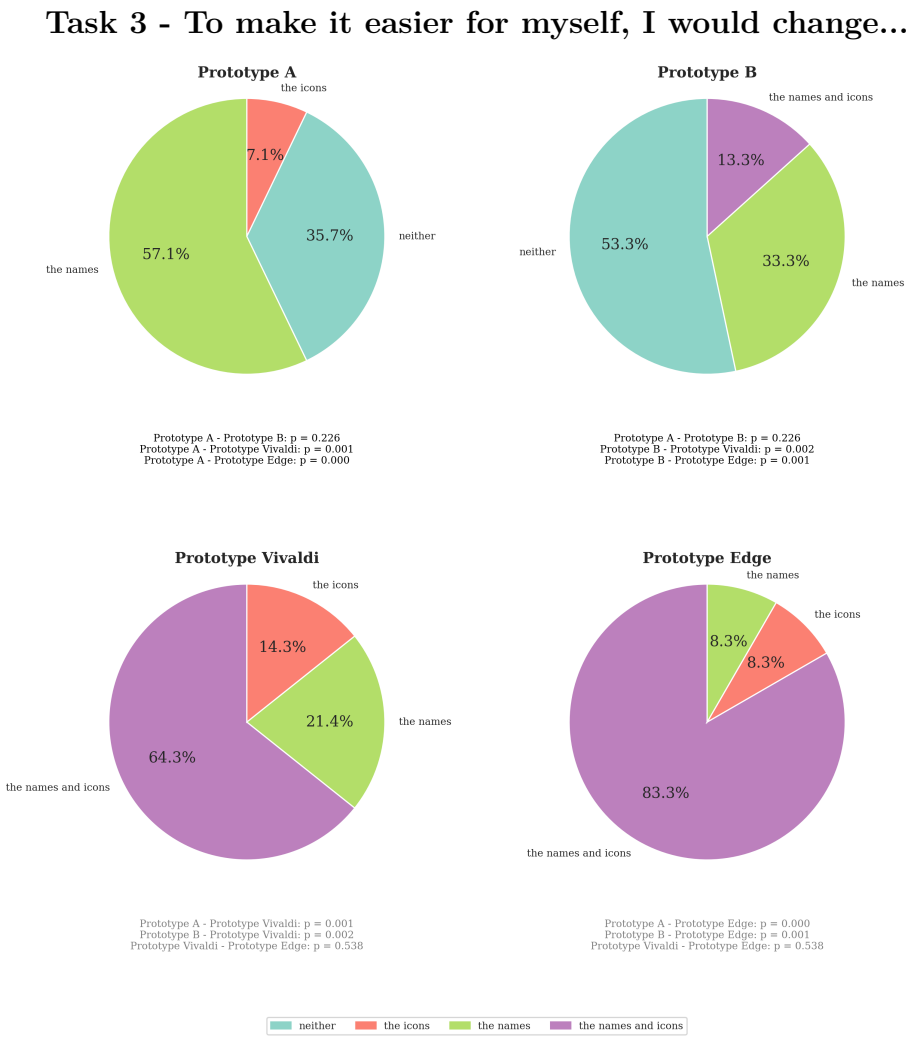


Figure 12.23: Chi-Squared Test of Independence indicated significant differences for Prototype A - Prototype Vivaldi ($p<0.001$), Prototype A - Prototype Edge ($p<0.001$), Prototype B - Prototype Vivaldi ($p=0.002$), and Prototype B - Prototype Edge ($p<0.001$); the remaining comparisons were not significant.

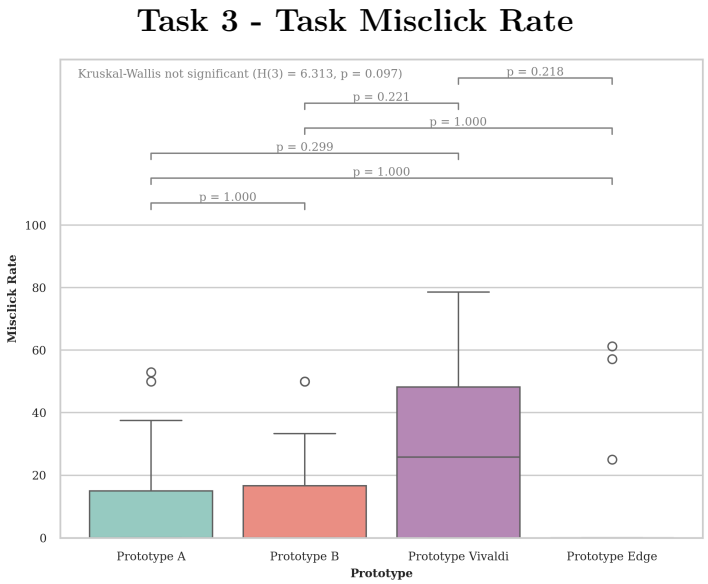


Figure 12.24: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=6.313, p=0.097), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

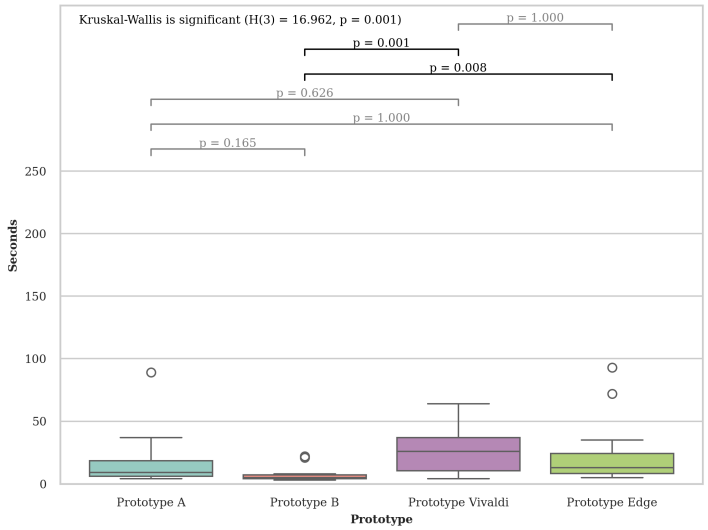


Figure 12.25: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=16.962, p<0.001), indicating significant differences among prototypes. Dunn post-hoc comparisons revealed significant differences between Prototype B - Prototype Edge (p=0.008), and Prototype B - Prototype Vivaldi (p<0.001); the remaining comparisons were not significant.

12.6 Task 4 - Switching between two previously saved windows

In this task, participants switched from one saved window to the other. They reported their preference for opening videos in the current versus a new window. Depending on which prototype they were using, the question was phrased differently. Participants using Vivaldi, Prototype A or Prototype B were asked: "I liked that my videos opened in the current window - instead of in a new window." Participants using Edge were asked: "I liked that my videos opened in a new window - instead of in the current window."
In figure 12.26, participants' ratings for opening videos in the current window versus a new one showed no statistically significant differences, even though Prototypes A and B scored slightly lower than Edge and Vivaldi.
Figure 12.27 shows the misclick rates when switching back to a previously saved window. All prototypes produced comparable misclick rates with no significant differences.
Figure 12.28 presents the time taken to switch between saved windows. Completion times were alike for Prototypes A, B, Edge, and Vivaldi, with no significant differences observed.

Task 4 - I liked that my videos opened in the current window - instead of in a new window. (alternatively: I liked that my videos opened in a new window - instead of in the current window.)

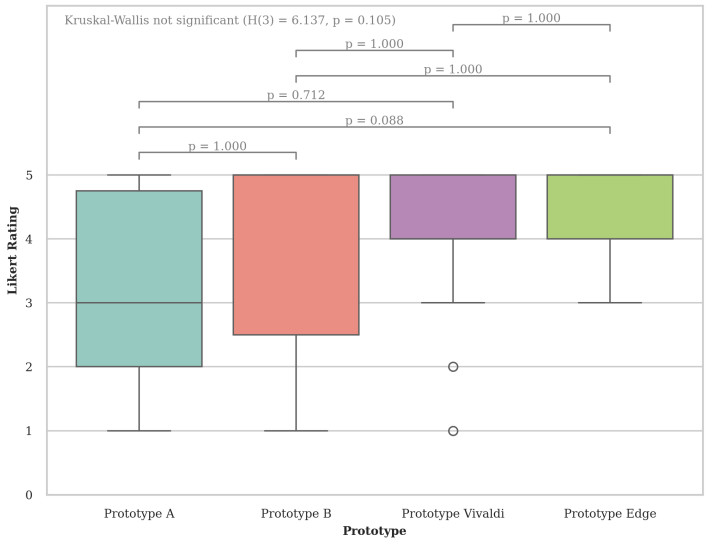


Figure 12.26: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=6.137, p=0.105), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

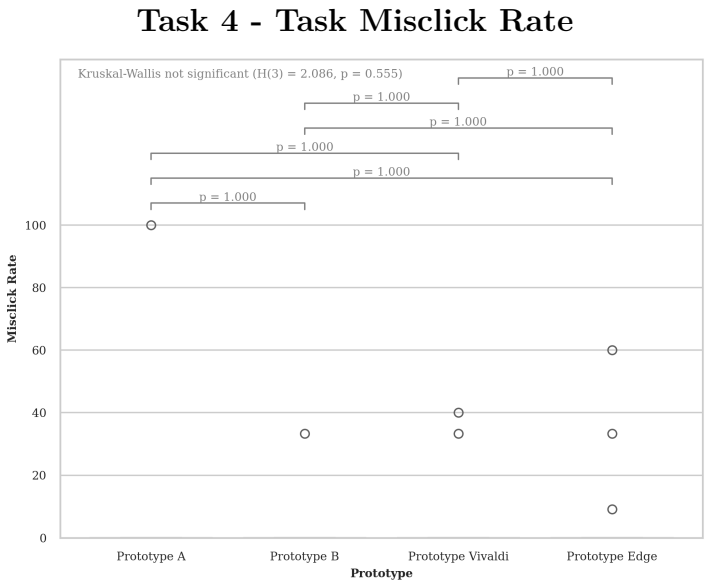


Figure 12.27: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=2.086, p=0.555), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

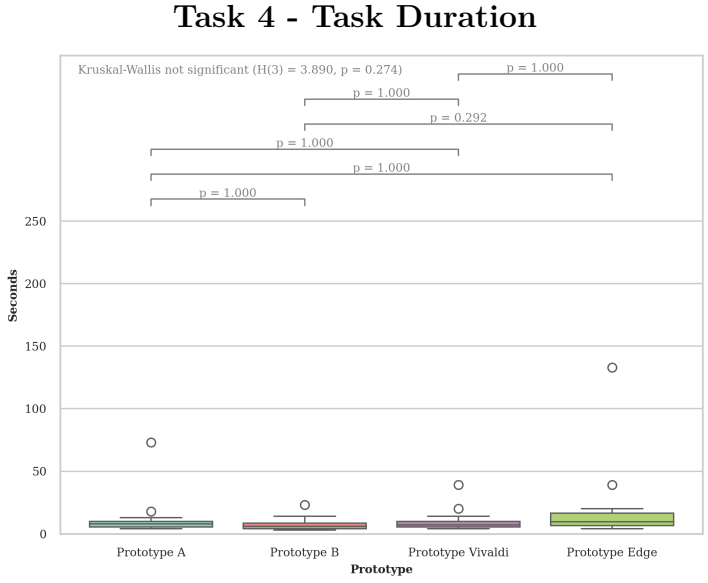


Figure 12.28: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(3)=3.890, p=0.274), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

12.7 Task 5 - Opening a saved window in a new window

In this task (not applicable to Edge, where saved windows always open in a new window), users were asked to open a previously saved window alongside the current

one. They rated the ease of discovering this option.
In figure 12.29, participants rated the ease of opening both windows at once. Scores were similar for all prototypes, with no statistically significant differences.
Figure 12.30 shows the misclick rates when users tried to open two separate windows. Misclick rates did not differ significantly between prototypes.
Finally, figure 12.31 presents the time taken to complete this task. Task durations were comparable across all prototypes, and no significant differences were observed.

Task 5 - Finding how to open both windows at the same time was easy for me.

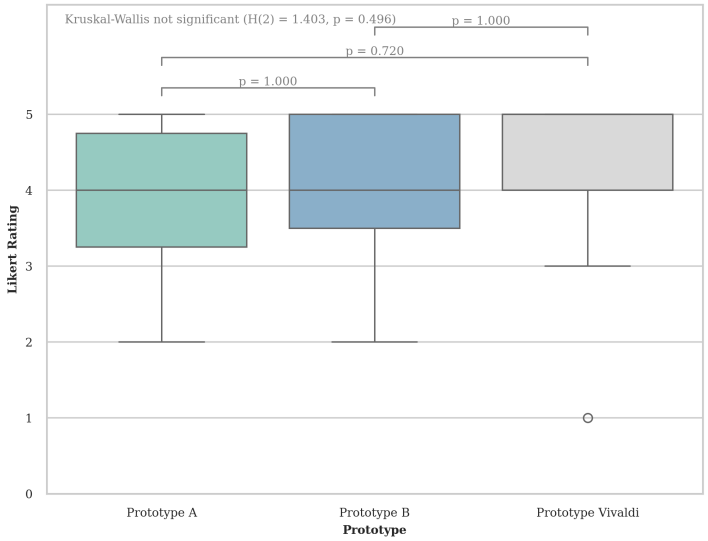


Figure 12.29: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(2)=1.403, p=0.496), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

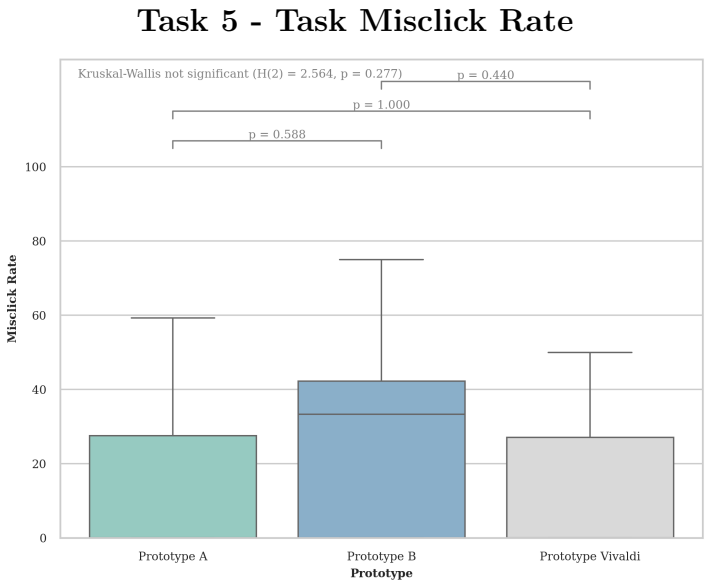


Figure 12.30: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(2)=2.564, p=0.277), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

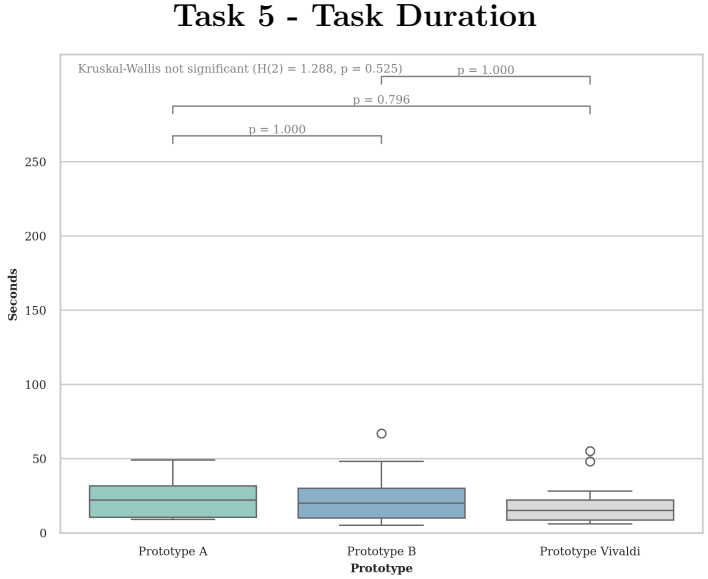


Figure 12.31: Likert scale ratings were analyzed using a Kruskal-Wallis test (H(2)=1.288, p=0.525), indicating no significant differences among prototypes. Dunn's post-hoc comparisons did not reveal any significant differences.

12.8 Analysis of Evaluation

Demographics and Metadata The imbalances in participant demographics suggest potential biases. An overrepresentation of Safari users in Prototype A could limit the generalizability of its results to other browsers. Participants in B and Edge

who report higher technical skills may complete tasks faster or make fewer misclicks, independent of interface differences. Meanwhile, the larger share of novices in A and B, who are unfamiliar with any window-saving tool, may be more prone to errors or slower task completion simply due to a lack of prior exposure rather than design shortcomings. Finally, the skew in gender for B and Edge, though not statistically significant, may also influence the results.

Aggregate Usability Results Prototype B showed the strongest overall performance, scoring higher on the SUS than Vivaldi, achieving better average task ratings than both Edge and Vivaldi and completing tasks more quickly than both browsers. It also had fewer misclicks than Vivaldi, though its misclick rate did not differ significantly from Edge. Prototype A outperformed Edge and Vivaldi on average task ratings but did not differ significantly from any browser in SUS scores, misclick rates, or completion times.

Task 1 and Task 3: Ease of Saving Windows When asked to rate how easy it was to save their window, both prototypes A and B significantly outperformed both Edge and Vivaldi on task 1, as can be seen in figure 12.11.
When asked to rate the same procedure after performing it a second time in task 3, both prototypes A and B outperformed Vivaldi significantly, as shown in figure 12.19. While Edge was rated higher than Vivaldi, the difference was not significant in comparison to Vivaldi or the other prototypes.
The time taken to complete tasks 1 and 3 was significantly lower for prototype B compared to Edge and Vivaldi; see figure 12.14 and figure 12.25. When comparing prototype A against Edge and Vivaldi, we see similar results, although they are not statistically significant.
Looking at the misclicks prototype, B significantly outperforms Edge and Vivaldi for task 1, as shown in figure 12.13. While not significant, prototype A also outperforms Edge and Vivaldi. When looking at the results for task 3, none of the differences are significant. However, Edge does seem to perform better compared to itself in task 1, which could indicate that Edge's design is more intuitive for users after they have used it once.
The differences between these results are quite pronounced, perhaps because prototypes A and B have much more visible and accessible save buttons for saving a window with open tabs, while Edge and Vivaldi make this feature less prominent. Edge and Vivaldi seem to assume that users will start new windows for new projects and rarely need to save their current progress, so they prioritise creating new windows over saving existing ones, which might be a misprioritisation, as users often find themselves in the middle of a new project that they wish to save rather than rigorously creating new workspaces when starting projects. There is also no negative impact on users who prefer to create saved windows before starting a project, as this feature is equally accessible in A and B as in Edge and Vivaldi.

Task 1 and 3: Confidence of Saving Windows When users were asked whether they felt confident that they had successfully saved the window, both prototypes A and B significantly outperformed Vivaldi, as can be seen in figure 12.12.

Edge performed slightly better than Vivaldi, though not significantly and not as well as A and B.
This suggests that the strategy of keeping the popup open and showing the just saved window inside the popup is effective in making users feel confident that they have successfully saved their window.
The difference between Edge and Vivaldi could be explained by how they indicate that the window was saved after closing their respective popups. Vivaldi changes the top left button's text from "Workspaces" to "New Workspace", while Edge replaces the icon with a text button labelled "Workspace 1" and changes the window's background colour to highlight the update. Edge has a more visible change, which could explain the difference in confidence between the users.
While the results are not significant, a similar pattern can be seen in task 3, see figure 12.20, where Vivaldi seems to have performed worse than the other prototypes. There is no discernable difference between prototypes A, B, or Edge.
The prominent save button and not closing the popup in prototypes A and B seem to assure users that they have saved their window. Edge's more visible change after saving the window also seems to work well. In comparison, users of Vivaldi are not confident that they have saved their window, despite having saved a window in this way before, which could indicate that Vivaldi is providing insufficient feedback to the user after saving a window.

Task 2: Confidence of Not Losing Tabs When Switching Between Windows While users seem to favour prototypes A and B slightly, the results are not significant; see figure 12.15.

Task 2 and 4: Preference Between Opening a New Window or Replacing The Current Window While the results are not significant, users seem to slightly prefer prototype Edge and Vivald over A and B; see figure 12.16 and figure 12.26.
Prototype A was inconsistent in how it handled opening a saved window compared to an unsaved window; this was intended to test whether users would accept the inconsistency as a tradeoff for potentially gaining the benefits of both systems. The results are insignificant and hard to interpret with any confidence.
Prototype B was consistent with itself and functioned similarly to Vivaldi in this regard. Still, B has performed worse than Vivaldi; it is unclear why this is the case, as the two systems were essentially identical. However, the difference is not significant and could be due to random chance.

Task 5: Opening a saved window in a new window Task 5 was mostly inconclusive, but it does not seem that users had any difficulty finding how to open a saved window in a new window in any of the relevant prototypes, see figure 12.29.

Satisfaction with Default Names and Icons of Newly Saved Windows When users were asked to rate how satisfied they were with the default names and icons of their newly saved windows, both prototypes A and B very significantly outperformed Edge and Vivaldi, see figure 12.21 and figure 12.22.

When asked whether they would like to change the names and/or icons of their newly saved windows, there were significant differences between prototypes A and B compared to Edge and Vivaldi, see figure 12.23. A clear majority of users wanted to change both the default names and icons in the Edge and Vivaldi prototypes while changing neither or just the name was very common in A and B.
The default names and icons of the saved windows in prototypes A and B are based on the title and icon of the last open tab in the window. Users seem to prefer this system over the default names and icons of Edge and Vivaldi, which are given generic names and icons. In Vivaldis' case, the icon is randomly selected from a set of icons, while Edge uses the same icon for all saved windows.
It is not unlikely that users would be confused if the names and icons of their saved windows suddenly changed, as they would in prototypes A and B. However, the data strongly suggests that users are not satisfied with the default names and icons of Edge and Vivaldi and would like to change them, which is not fully supported in either system.

Summary of Evaluation Analysis Across the overall measures, Prototype B performed best, with higher SUS scores, better average task ratings, faster completion times, and fewer misclicks compared to the standard browsers. Prototype A also earned higher task ratings than Edge and Vivaldi but did not differ significantly in SUS, misclick rate, or speed.
Prototypes A and B generally performed better than Edge and Vivaldi when it came to saving windows.
Prototypes A and B provided a more visible and accessible save function, resulting in faster task completion, fewer misclicks, and increased user confidence.
In addition, users expressed higher satisfaction with the default names and icons in prototypes A and B, preferring them over the more generic or inconsistent alternatives found in Edge and Vivaldi.

13
Discussion

Insights from Competitor Analysis The results indicate that Firefox lacks important task and tab management features that users want and need. Edge and Vivaldi provide built-in workspace management systems designed explicitly to help users save, organise, and resume tasks effectively. While Firefox extensions, such as STG, can sometimes provide mostly satisfactory alternatives to these features, they pose several issues: they are harder to discover, less refined, potentially less trustworthy, and present risks of data loss or security concerns. Therefore, native solutions would likely increase usage and provide a more secure and reliable user experience.

Insights from Task Analysis The task analysis showed that Firefox's existing methods of preserving browsing states, keeping windows open, bookmarking tabs, and restoring recently closed windows impose significant cognitive and organisational loads on users. Despite a clear need for a solution, none of the methods currently available in Firefox are intended for managing browsing sessions.
Other browsers offer solutions that let users save their windows using workspaces, and the STG extension provides Firefox with similar functionality.
The task analysis also showed that designing these solutions with fewer steps required for users to save their current windows is possible since both Edge and Vivaldi optimise for creating new windows instead of saving current ones.

Insights from Workflow Analysis The workflow analysis revealed significant room for variation in user workflows. While saving and resuming tasks likely benefit users of all possible workflows by preserving progress, customisation through naming and visual identification significantly enhances ease of use and organisation.
Expert features, such as keyboard shortcuts or experimental solutions to niche issues, like workspace linking, are also worth investigating. Collaboration and multi-device synchronisation will likely become essentials of future task management tools. Finally, allowing users to organise saved windows into folders is expected to be useful, particularly for managing multiple or complex tasks.
Additional research should explore these features, assessing user demand, potential use cases, and the complexity these features might introduce.

Insights from Analysis of User Needs The user needs analysis showed that users prefer tools that preserve the state of their browsing sessions without introducing additional complexity. Workspaces should facilitate users' existing habits rather than disrupt them.

Users should feel confident that their tabs are secure and that their saved progress is safe and secure. They should also be able to easily restore windows lost through crashes or human error. It would also be helpful to introduce visual feedback when users create and delete workspaces. Additionally, incorporating an undo option allows users to quickly reverse any deletions they may regret. Another solution could be an archive for unsaved or deleted windows, allowing users to recover them later.
Visual identification methods, such as customisable window names, window icons, and window colours, could help users locate saved sessions more quickly. However, simplicity remains important; while options for naming and organising workspaces are beneficial, they should remain optional, allowing users to decide the degree of organisation they prefer.
In addition, the interviews show that users do work on multiple screens and devices, demonstrating the need for cross-device synchronisation and adaptable workflows.
Automatic organisational features could also be valuable, provided users retain control over them. However, the usefulness and intrusiveness of automation features need further examination.

Insights from Analysis of Usability Test Results Usability testing revealed that users easily found the "Create New Workspace" command but struggled to locate the "Save Window" function in Edge and Vivaldi.
Users were positive towards customising windows with colours and icons, likely because it makes workspaces feel more personal and easier to distinguish at a glance.
Users also preferred opening new windows only when requested rather than automatically when opening workspaces, like in Edge. However, these tests were conducted without prior experience with the tools, meaning users might adapt or feel differently in actual practice. To address this issue, choose one option as the default and place the other in a context menu, similar to Vivaldi. Allow users to customise which option is primary and which is in the context menu.

Insights from Analysis of Evaluation The evaluation makes it clear that making the save function prominent pays off. In both Task 1 and Task 3, prototypes A and B, with their large, clearly labelled save buttons, allowed participants to save windows more quickly, with fewer misclicks, and with greater confidence than Edge or Vivaldi. Edge users improved on their second try compared to Vivaldi but still lagged behind the performance seen in A and B. These results show that a prominent and labelled save button will improve usability if saving windows, as opposed to creating new workspaces, is important.
The difference in confidence between Edge and Vivaldi users in task 3 can possibly be explained by how each browser signalled that the window was saved, where Edge shifts the window's background colour in addition to updating the name and icon in the workspace button, making the change far more visible.
Default names and icons greatly influence usability. Prototypes A and B set the workspace titles and favicons from the last active tab. They received higher satisfaction ratings than Edge's and Vivaldi's default names, icons and colours. Most Edge and Vivaldi users wanted to rename or change the icons of their saved windows

for easier identification. In contrast, few users felt that need with A and B. Clear and meaningful defaults help users quickly identify their workspaces correctly. A caveat of these results is that the prototypes were largely static, which likely benefits prototypes A and B since it is unclear how users would react if their workspace name and icon changed based on which tab they used last. Another solution could be to set the name and icon as the window's current icon and title once when the window is saved and not update it afterwards. Either way, some implementation which produces better default names and icons is likely preferable.
Preferences for opening behaviour were unclear. Ratings on workspaces "opening in a new window" versus "replacing the current window" showed no significant differences. Finally, the participant sample had uneven gender, browser, and skill distributions, as well as small group sizes. These factors suggest that the findings should be viewed as indicative rather than conclusive. Still, the advantage of visible, informative save controls and meaningful defaults is clear.

14
Conclusion

What do users need to manage browser tabs, windows and tasks effectively? Users need a reliable way to save complete browsing sessions, including navigation history, last open tab, scroll position, and other state preservation context, as well as reassurance that their tabs are secure and can be recovered if lost (peace of mind). They want the controls for this to be obvious and quick (ease of use), with previews, names or colours that help them spot the right tab or window at a glance (visual cues). Because people arrange their work differently, the tool must allow them to group, split, and shuffle tabs however they like (flexible workflows) and capture work that starts informally, such as saving their current window where they already have open tabs without extra setup (support for unstructured work). Users should not feel like the organisation is being forced upon them, for example, by requiring them to name windows (non-intrusive organisation). The result should be a workspace that helps them stay uncluttered (tidy workspace), offers easy overviews or reminders, and syncs smoothly across devices for uninterrupted flow (cross-device sync).

Does Firefox support these needs? Firefox supports keeping windows open, bookmarking tabs, and restoring recently closed windows. However, none of them provides a way to save an entire window, including its tab titles, icons, or previews. These methods require extra steps and fail to preserve the entire state of the window, or they increase the cognitive load by forcing users to keep windows open instead of saving, closing, and reopening them. Therefore, Firefox only partially fulfils users' core needs.

Can a Firefox extension meet these needs? Extensions such as Simple Tab Groups let users save, close, and resume Windows. STG proves that most workspace functionality can be achieved in add-ons. However, extensions depend on user discovery, carry security and data-loss risks, and cannot integrate as deeply as native code, which prevents some of the visual cues present in Edge and Vivaldi. For example, while it is possible to create an extension that changes the button for opening workspaces to display the icon of the currently open workspace, it is not possible to display the workspace name as both Edge and Vivaldi do. Therefore, while an extension can do the job, it cannot guarantee the usability, reliability, and broad adoption that a built-in solution would provide.

How can window-management tools (workspaces) be improved? The study indicates that the following would improve workspaces:

- **Emphasise saving windows over creating workspaces.** Emphasise saving windows rather than creating new workspaces, and include a clearly labelled "Save window" button.
- **Prevent real and perceived loss of data.** Prevent data loss and include recovery options while also implementing features and visual cues that allow users to feel safe and in control. For example, provide visual feedback and undo options when deleting windows, as well as an archive of previous windows.
- **Use meaningful visual cues.** Present workspaces using name, icon, colour, and possibly thumbnail previews. Ensure the default titles and icons represent the workspace, for example, by setting them to match the title and favicon of the last active tab.
- **Flexible window opening behaviour.** Let users choose whether a workspace opens in the current window or a new one, with easy settings to switch defaults.
- **Advanced features for power users.** Offer workspace folders, keyboard shortcuts, cross-device sync, collaboration tools, opt-in automation (e.g., auto-sorting) that can be turned off, and workspace linking for managing complex, multi-window tasks. These features must not complicate the primary workflow of saving and resuming windows.

Implementing these improvements would give users a trustworthy and straightforward way to save, organise, and resume tasks.

15
Recent Developments

As of April 2025, two browsers have released updates to their workspace and tab-management interfaces that align with the recommendations of this study.

Edge: Prioritising "Save Window" Edge has replaced the `+ add icon` button for creating new workspaces and reworked the `··· horizontal three-dot icon` menu that previously hid the "Save all tabs as a workspace" command. In the new version, the `+ add icon` icon is now a `⌄ dropdown icon` dropdown labelled "New." Clicking it reveals two options: "Save tabs to a workspace," then "Create a workspace." By making this change, Edge has made the option to create a new workspace take one extra click, and in return, they have made saving the current window equally important as the option to create new workspaces. Since the "Save tabs. . . " option is listed above the "Create. . . " option, saving windows seems to be presented as a higher priority than creating workspaces.
This interface change reinforces one of the key conclusions of this study that the "Save window" action should be prioritised over creating new workspaces. The new design in Edge makes it clear that Microsoft is aware of the importance of this feature and is actively working to improve it.

Firefox: Introduction of Tab Groups Firefox now offers a native Tab Groups feature, allowing users to organise tabs into named groups on the tab bar. This addition addresses some of the organisational needs identified in this study, but Firefox still cannot save and restore complete window states.

Firefox: AI-powered group naming (opt-in) Firefox has also added a "Name with AI" button to the context menu of each tab group. When clicked, the browser reads the titles of the tabs in the group to an AI model. The model returns a short suggestion such as "Travel research" or "Budget report", which the user can accept, edit, or ignore. Because the feature is activated only when pressed, it follows this study's recommendation that automation should be opt-in and under full user control.

16
Future Work

This chapter outlines ways to explore and refine workspaces beyond the current study's scope.

Larger Evaluation Conducting a larger evaluation with more participants would provide a better understanding of the impact of workspaces on user experience. The study could include a broader range of users, tasks, and environments to assess the generalizability of the findings.

Long-Term Evaluation Conducting a long-term study in real-world settings would improve the validity of the findings. This study could uncover adoption and workflow patterns that are not clear in controlled conditions, showing how these patterns appear in daily work routines and interactions.

Broader Competitor Analysis Expanding the analysis to include a broader range of competitors could provide additional insights into effective workspace management strategies. This could reveal innovative features or approaches that could inform future development.

Static vs Dynamic Workspace Names Two improved default naming strategies remain unexplored: fixing the name after the first save (static) and updating it automatically to match the active tab (dynamic). Other naming strategies might be possible and should be assessed. These evaluations should be compared to the simple default naming method, "Workspace #."

Advanced Interaction Techniques Future studies could investigate keyboard shortcuts, mouse gestures, and command-palette actions for frequent operations such as saving, renaming, and switching.

Hierarchical Organization Creating folders would help users stay organised when the number of workspaces grows. Future studies could compare using named folders to manual naming and sorting workspaces.

Linked Workspaces Allowing multiple workspaces to open with a single click would make it faster to launch a predefined set of windows, which could be particularly helpful for expert users. This could be achieved by allowing the user to open entire folders of windows simultaneously or link workspaces so that they open

together. Future research could explore other options, look at similar tools for inspiration, and assess the demand for or benefits of such features.

Cross-Device Sync Syncing information across different devices significantly benefits users who switch between multiple computers. Future studies could conduct interviews with users to identify potential use cases and later perform a larger study on usability.

Collaboration Creating shared workspaces could offer significant benefits for users who work in teams. Future studies could investigate the user needs of workspace collaboration at different scales, from two-person collaboration to large teams and even organisations.

Window Reuse Behaviour Opening a workspace in the current window versus a new window remains a design trade-off, and the results are largely unknown. Allowing the user to set a default behaviour could also be tested in a longer-term study. A future study could compare user satisfaction with the different modes to see if the average user has a preference.

AI Assistance Some participants suggested automated assistance in managing workspaces, tabs, and windows. Recent developments in AI have opened many new possibilities; perhaps the user can click a button to get suggestions for workspace organisation or naming, automatically grouping their tabs, or renaming them based on their content. Future work could explore these areas extensively and investigate the usability of specific existing features, such as the newly added AI group naming feature in Firefox.